UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Diseño e Implementación de un Sistema basado en Android y Bases de Datos NoSQL para el Seguimiento de Pacientes

Máster en Computación Paralela y Distribuida

Departamento de Sistemas Informáticos y Computación

Universitat Politècnica de València

Curso Académico 2014 - 2015

Trabajo Fin de Master

Autor

David Arce Grilo

Director

Ignacio Blanquer Espert

Valencia, 18 de Septiembre de 2015

Resumen

El auge de las aplicaciones distribuidas y las tecnologías móviles en los últimos años está suponiendo una oportunidad para algunos campos de la medicina donde uno de los principales problemas es la falta de adherencia al tratamiento, desencadenando recaídas y re-hospitalizaciones. Estas tecnologías propician el desarrollo de aplicaciones de monitorización continuada que pueden mitigar dichos problemas. No obstante, es importante analizar el grado de madurez de las tecnologías actuales para entornos realistas donde aspectos como la minimización de comunicaciones es fundamental. El objetivo del proyecto es proponer y evaluar un diseño de aplicación que facilite el seguimiento de la actividad y el estado de personas a través de dispositivos móviles y sus tecnologías. Para ello, se ha creado un sistema de monitorización y recolección de datos para pacientes psicóticos basado en una aplicación móvil que le recuerda la toma de medicación y le formula diversas preguntas acerca de su estado y de los efectos de la medicación. La visualización de estos datos de seguimiento por parte del médico se realiza vía web, teniendo una visión permanentemente actualizada del estado del paciente. Las pruebas realizadas han mostrado que el sistema es fiable, consume pocos recursos y se comporta de manera robusta en entornos de alta demanda. En conclusión, podemos afirmar que las tecnologías utilizadas en este proyecto son adecuadas para satisfacer las necesidades encontradas.

Abstract

The rise of distributed applications and mobile technologies in recent years is opening an opportunity for some fields of medicine where one of the main problems is the lack of treatment adherence, brining relapses and new hospitalization episodes. These technologies encourage the development of continuous monitoring applications that can overcome these problems. However, it is important to analyze the maturity of the existing technologies for realistic environments where aspects such as the minimization of communications are critical. The aim of the project is to design and evaluate an application in order to monitor a patient's activity and status through mobile devices and related technologies. Therefore, a monitoring and data collection system has been developed for psychotic patients based on a mobile application that reminds taking medication and makes a couple of questions about his/her condition and the medication effects. The monitoring data display is done by the physician via web, having a continuously updated view of the patient's condition. The tests have shown that the system is reliable, requests few resources and performs robustly in highly demanding environments. In conclusion, we assess how the technologies used in this project fit the needs.

Índice

1. Introducción

1.1 Motivación

En los últimos años hemos visto como las tecnologías móviles han evolucionado de forma exponencial. Hoy en día hay más de 7.000 millones de abonados a la telefonía móvil en el mundo, cifra que el año 2000 era de 748 millones. El crecimiento de usuarios del *smartphone* está creciendo tres veces más rápido que la tasa de ordenadores, y desde 2012 que lo superan en número. De manera similar, entre 2000 y 2015 la penetración de Internet se ha multiplicado por siete, pasando de 6.5 al 43 por ciento de la población mundial. Además, la conexión a Internet mediante dispositivos móviles está creciendo mucho más rápido que la conexión de banda ancha desde dispositivos fijos.

Esto genera una oportunidad para algunos campos que no aprovechan completamente este tipo de herramientas móviles, como en el campo médico en el tratamiento de pacientes de psicosis. La psicosis es una condición mental severa definida como una pérdida de contacto con la realidad. Los síntomas característicos de la psicosis son, por un lado, las alucinaciones, que corresponden a una percepción errónea e inconsciente del entorno, como la percepción de un proceso sensitivo en ausencia de una causa externa, y por otro, los delirios, que corresponden a una interpretación errónea de la realidad y se manifiestan en forma de ideas y creencias falsas resistentes a la autocrítica por parte del paciente.

Los pacientes que reciben tratamiento antipsicótico por vía oral tienen una adherencia estimada de menos de un 60% [1] [2] y un 70% de los pacientes psicóticos abandona el tratamiento a los 6 meses de iniciarse [3] [4]. La tasa media de adherencia con medicación antipsicótica es de un 58%, con un rango variable desde el 24 al 90%. El estudio CATIE [3] llevado a cabo con enfermos esquizofrénicos subraya que éstos a menudo abandonan el tratamiento o no lo cumplen independientemente de la medicación que reciben, de modo que un 74% pacientes interrumpen el tratamiento antes de los 18 meses y un 40% de los pacientes interrumpen el tratamiento por voluntad propia. La falta de adherencia da lugar a un aumento del número de recaídas y re-hospitalizaciones, que en muchos casos se vuelven recurrentes. De este modo podemos considerar la falta de adherencia al tratamiento antipsicótico como una causa de resistencia al mismo, constituyendo un verdadero reto para el psiquiatra en el manejo de estos pacientes.

Las aplicaciones móviles permitirían establecer un canal bidireccional por el cual se mejoraría la adherencia al tratamiento y permitirían introducir cambios en dicho tratamiento o en la monitorización a medida que evoluciona un paciente entre las visitas presenciales del paciente. Para ello, hay que diseñar una arquitectura reactiva que facilite dicha comunicación de forma eficiente.

1.2 Objetivos

El objetivo del proyecto es proponer y evaluar un diseño de aplicación que facilite el seguimiento de la actividad y el estado de personas a través de dispositivos móviles y sus tecnologías, y demostrarlo sobre una aplicación de monitorización del estado clínico de un paciente para la mejora de la adherencia al tratamiento médico.

Para ello se debe diseñar:

- Una **aplicación móvil** basada en Android que recopile información de seguimiento mediante notificaciones programadas con cierta frecuencia, y las envíe a una base de datos. Debe comportarse de manera reactiva, detectando cambios en la configuración y reprogramando así el horario de notificaciones. También hay que tener en cuenta los recursos de memoria empleados, el gasto en batería, y diseñar una interfaz amigable y que requiera la mínima interacción posible con el paciente para facilitar su uso.
- Una **base de datos** donde se almacene tanto la información recibida por el paciente como la configuración de programación de las notificaciones. La base de datos debe poder contener una gran cantidad información sin comprometer su velocidad y ser eficazmente escalable.
- Una **aplicación web** donde el médico pueda visualizar en cualquier momento y lugar la información de seguimiento de sus pacientes, además de establecer la configuración de las notificaciones del paciente. Esta web debe ser reactiva y autoactualizarse si llega algún dato de seguimiento nuevo, ser compatible con los principales navegadores web, y disponer de una representación de datos cómoda y de fácil interpretación para el médico.

Por otra parte, se deben realizar pruebas de carga para comprobar la robustez del sistema y medir sus prestaciones y consumo de recursos, y pruebas con médicos y pacientes reales para evaluar la satisfacción global del trabajo.

1.3 Estructura del Documento

En primer lugar, en el estado del arte se expondrá como el auge de los dispositivos móviles está ayudando a ser la herramienta idónea para poder monitorizar y realizar un seguimiento de los pacientes que se encuentren bajo tratamiento médico, mejorando también la adherencia al mismo. Existen varios estudios sobre el uso de los dispositivos móviles tanto en el campo de las patologías médicas, como en el de la salud mental o, más concretamente, en el de la psicosis.

También como parte del estado del arte, se listarán las diferentes tecnologías que se han requerido para la implementación del proyecto, tanto en el campo de las aplicaciones móviles, como en el de las bases de datos y en el de las aplicaciones web. Se describirán brevemente y se enumerarán sus principales características.

En la parte del diseño, se empezará describiendo en qué consiste concretamente la realización del seguimiento: qué datos se espera recopilar y como serán mostrados.

A continuación, se mostrarán las tres partes en las que se divide el sistema: aplicación móvil, base de datos y aplicación web. Se analizarán como los actores del sistema interactúan con cada parte y se detallará el funcionamiento de cada una de ellas.

Se empezará por la base de datos, definiendo su estructura y explicando cómo esta puede cambiar y crecer a lo largo del proceso de seguimiento.

En cuanto a la aplicación móvil, primero se definirán cuáles son las tareas que esta debe realizar. Después, los conceptos y mecanismos del entorno Android que se han empleado para el proyecto y que son necesarios comprender para abordar la explicación del diseño. Finalmente, se detallará el diseño, mostrando como se componen las diferentes partes del sistema, como funcionan y cómo interactúan entre ellas.

Como último punto del diseño, se explicarla el funcionamiento de la aplicación web. De manera similar, se definirá las tareas que debe realizar, se explicarán los conceptos y mecanismos empleados y se detallará el funcionamiento de los componentes que se utilizan.

El siguiente paso será la implementación del prototipo. Primero se nombrarán las aplicaciones que se han utilizado, como el entorno de desarrollo, y se explicará el funcionamiento de los prototipos, tanto de la aplicación móvil como de la aplicación web, repasando cada una de las partes que los componen.

En el siguiente capítulo se abordarán las pruebas realizadas al sistema. En primer lugar, pruebas de carga para estudiar el rendimiento y la robustez de nuestras aplicaciones, y después, pruebas con usuarios reales que probarán las aplicaciones durante un periodo de 2 meses.

Por último, se mostrarán las conclusiones que hemos sacado de este proyecto y una serie de mejoras de cara a futuras versiones.

2. Estado del Arte

2.1 Uso de Dispositivos Móviles en el Seguimiento de Pacientes

2.1.1 Auge de la Tecnología Móvil

La infraestructura de las tecnologías móviles se ha desarrollado de forma exponencial a lo largo de los años. Más de un 90% de la población mundial tiene acceso a señal de telefonía móvil, traduciéndose en más de 7.000 millones de subscripciones en todo el mundo [5]. Debido al elevado crecimiento de los usuarios de *smartphone*, estos dispositivos superaron a los ordenadores en 2012. Como se pueden ver en las Figuras 1 y 2, actualmente la instalación del *smartphone* está creciendo tres veces más rápida que la de ordenadores, y la conexión a Internet mediante dispositivos móviles está creciendo mucho más rápido que la conexión de banda ancha desde un dispositivo fijo.

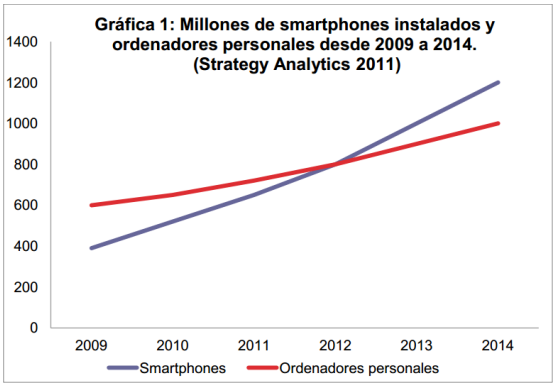

Figura 1. Smarthphone vs. Ordenador Personal

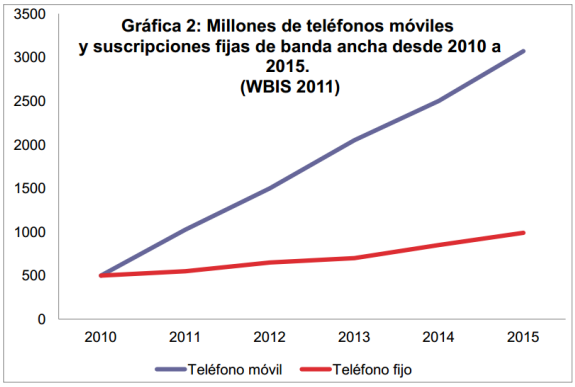

Figura 2. Teléfono Móvil vs Teléfono Fijo

2.1.2 e-Health y m-Health

Las tecnologías *e-Health* combinan el uso de la comunicación electrónica y TICs con usos clínicos, educativos, éticos y administrativos con el objetivo de mejorar el sistema sanitario, promocionar la salud, y permitir un mayor acceso sanitario a toda la población. El Observatorio Global para la Salud (GOe) define *m-Health* o *mobile Health* como aquella práctica médica y de salud pública cuyo soporte electrónico corresponde a un dispositivo móvil, de seguimiento, una PDA y otros dispositivos provistos de tecnología wifi [6].

2.1.3 Las Aplicaciones Móviles en el Sistema Sanitario

De esta forma, la relación entre paciente y médico está cambiando y aparece la figura de "paciente empoderado". Desde el punto de vista del paciente, el empoderado es una persona con información y conocimiento de su enfermedad, del tratamiento de la misma y con poder para decidir sobre ella. En este sentido, cabe destacar el aumento de las consultas a Internet por parte de los pacientes, de modo que el 72% de los internautas españoles consulta sus dudas de

salud en Internet. [7], siendo el autodiagnóstico su principal uso, en el 33% de los casos. La revolución en las TICs que permiten la monitorización del paciente, constituyen un modelo de atención centrado en su automanejo y en las intervenciones simultáneas por parte del personal sanitario. Los desafíos para percatarse de la existencia de esta visión incluyen una madurez tecnológica, privacidad, seguridad, y la habilidad del médico para personalizar soluciones que puedan maximizar el compromiso del paciente así como un cambio de comportamiento respecto a su patología. [8].

Algunos ejemplos de uso de aplicaciones móviles en el ámbito sanitario son: la promoción de hábitos saludables (adherencia a la medicación, dieta y ejercicio) y como complemento adicional de las intervenciones. Ya han sido utilizadas en patologías como crónicas como la hipertensión arterial, diabetes, la quimioprofilaxis para la malaria o el tratamiento antirretroviral del virus de la inmunodeficiencia humana (VIH) [9] [10] [11] [12]. Sin embargo, muchos ensayos de intervenciones móviles en enfermedades orgánicas evidencian una gran falta de participación, sugiriendo que estas tecnologías, a menudo, se encuentran poco integradas en la práctica clínica.

También existen algunos ejemplos recientes del uso de la tecnología móvil para el control del abuso de alcohol y otros tóxicos [13] [14], tratamiento del abuso de tabaco [15], de la enfermedad mental severa [16] y para el control de distintos problemas como trastornos emocionales, la obesidad infantil, el cáncer o el dolor crónicos [17] [18] [19] [20].

Durante la pasada década el uso de las tecnologías para la monitorización de la psicosis ha ido en aumento [21]. En la mayor parte de las intervenciones, los pacientes completaban un cuestionario de autoevaluación varias veces al día [22]. Uno de los primeros conocimientos en este campo se basó en el PDA [23] [24], pero se están convirtiendo cada vez más en aplicaciones obsoletas, y ahora ocupan un lugar muy limitado en el mercado. En un estudio [25] se observa que el 83% de la muestra con psicosis posee un teléfono móvil, sugiriendo que su uso en los pacientes psicóticos es comparable con el de la población general.

Todos estos estudios revelan que el seguimiento continuado mediante aplicaciones móviles mejora notablemente la adherencia al tratamiento y sus resultados finales, corroborando la viabilidad del objetivo principal de este trabajo.

2.1.4 Motivación

A pesar de todo lo expuesto anteriormente quedan muchas incógnitas por resolver. En primer lugar, todavía no sabemos cómo se podrían reducir la intensidad de las intervenciones terapéuticas convencionales (el número y la duración de las sesiones) gracias a estas nuevas posibilidades de tratamiento, y hasta cuanto se podría llegar a reducir el coste sanitario. En segundo lugar, el tipo de herramientas y modalidades de dispositivo que podrían ser más eficaces. En tercer lugar, existen pocas intervenciones específicas que hayan demostrado un éxito considerable en la mejora de la adherencia. Por tanto debería de despertarse un interés significativo en el desarrollo de las intervenciones destinadas a la mejora de la adherencia de la medicación en los trastornos psicóticos [26] [27]. En cuarto lugar, una aproximación más exhausta correspondería a la monitorización de los pacientes durante el transcurso de su vida diaria. Una evaluación de los síntomas en tiempo real supone una atractiva apuesta mediante la cual se pueden llevar a cabo intervenciones inmediatas, y prevenir el deterioro del estado

mental del paciente. En quinto lugar, que repercusiones positivas y negativas tiene el uso de dispositivos móviles en la vida diaria del paciente psicótico, y finalmente cuáles serían las repercusiones sobre los síntomas y como éstos se verían reflejados por el paciente a través de estas herramientas para que el terapeuta pudiera detectar e intervenir en el manejo.

2.2 Base Tecnológica

El objetivo de este proyecto es diseñar y evaluar una aplicación que facilite el seguimiento de la actividad y el estado de personas a través de dispositivos móviles y sus tecnologías. Para ello, diseñaremos e implementaremos un sistema de monitorización de seguimiento de un paciente durante el transcurso de su tratamiento médico mediante una aplicación móvil, a la vez que desarrollar una aplicación web para que el médico pueda seguir los progresos del paciente.

Con este objetivo en mente, necesitamos las siguientes tecnologías:

Base de Datos. Necesario para el almacenamiento de los datos recopilados de los pacientes y la configuración de las notificaciones. Esta debe ser escalable, eficiente y flexible. Se propone el uso de bases de datos NoSQL por su flexibilidad estructural y por su escalabilidad y MongoDB por su penetración en el mercado.

Aplicación Móvil. El usuario se comunicará a través de su dispositivo móvil, por lo que es preciso desarrollar una aplicación móvil nativa que permita su funcionamiento autónomo y aproveche las capacidades de notificación del sistema. Utilizaremos Android como sistema operativo móvil por su alta implantación y facilidad de distribución de las aplicaciones.

Servidor Web. Necesario para albergar la aplicación web. Esta debe satisfacer una alta demanda de peticiones y ser compatible con las bases de datos MongoDB. Haremos uso de Node.js, ya que tiene una arquitectura asíncrona orientada a eventos, además de tener una cuota de mercado creciente en los últimos años.

Framework Web. Necesitamos un *framework* que nos ayude a crear una aplicación web de alto rendimiento, reactiva e integrada con MongoDB y Node.js. Meteor satisface estas necesidades ya que ha sido creada específicamente para facilitar el desarrollo en este tipo de entornos. Además, como parte del desarrollo web, también se hará uso de los lenguajes **HTML**, **CSS**, **Javascript** y **jQuery**.

Pasemos a describir cada una de estas tecnologías con más detalle.

2.2.1 Bases De datos NoSQL

El término NoSQL [28] (que significa “no solo SQL”) se refiere a un nuevo tipo de base de datos que no se ajusta al modelo de bases de datos relaciones. Estas no garantizan la propiedad ACID (“Atomicity, Consistency, Isolation, Durability”) ya que priorizan otro tipo de aspectos como la escalabilidad, el rendimiento o la flexibilidad de su estructura.

Entre sus características, se puede destacar las siguientes ventajas con respecto a otro tipo de bases de datos:

- Permiten manejar volúmenes de información mucho más grandes con una velocidad de acceso rápida.
- Facilitan la escalabilidad horizontal ya que son fáciles de usar en *clusters* de balanceo de carga convencionales
- Poseen estructuras de datos flexibles permitiendo añadir nuevos campos de forma dinámica o efectuar una migración sin tener que ser reiniciadas o paradas.

Por otra parte, las grandes diferencias que existen entre las bases de datos relacionales y las no relacionales son:

- No usan SQL como lenguajes para realizar las consultas. La mayoría de las base de datos NoSQL tienen formas distintas de realizar las consultas por medio de algún lenguaje de apoyo, por ejemplo MongoDB utiliza JSON y Cassandra utiliza el lenguaje CQL.
- Usan una arquitectura distribuida de forma nativa. Con este tipo de arquitectura se nos permite tener la información de forma compartida entre varias máquinas.
- No permiten operaciones de JOIN. La mayoría de las base de datos NoSQL al manejar grandes volúmenes de información la realización de un JOIN es algo costoso por lo cual la forma de hacerlo es mediante el *software*.
- No existe un estándar para todas. La gran diferencia con las bases de datos relaciones radica existen varias formas de almacenar la información (orientadas a clave-valor, a columnas, a documentos, a grafos).

Para este proyecto se ha considerado que las bases de datos NoSQL satisfacen las necesidades del mismo. Más concretamente, MongoDB.

2.2.2 MongoDB

MongoDB [29] es un sistema de base de datos multiplataforma orientado a documentos de esquema libre. Cada registro o conjunto de datos se denomina documento. Los documentos se pueden agrupar en colecciones, las cuales se podría decir que son el equivalente a las tablas en una base de datos relacional (sólo que las colecciones pueden almacenar documentos con muy diferentes formatos, en lugar de estar sometidos a un esquema fijo). Se pueden crear índices para algunos atributos de los documentos, de modo que MongoDB mantendrá una estructura interna eficiente para el acceso a la información por los contenidos de estos atributos.

Los distintos documentos se almacenan en formato BSON, o “Binary JSON”, que es una versión modificada de JSON que permite búsquedas rápidas de datos, y es de uso común en la programación web, sobretodo en JavaScript.

Esto resulta especialmente interesante para proyectos web como Node.js.

MongoDB está escrito en C++, por lo que es bastante rápido a la hora de ejecutar sus tareas. Además, está licenciado como GNU AGPL 3.0, de modo que se trata de un *software* de licencia libre.

2.2.3 Android

Android [30] [31] es un sistema operativo inicialmente pensado para teléfonos móviles. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma. Actualmente, Android es ampliamente utilizado en multitud de dispositivos como móviles, tabletas, televisores o sistemas empotrados.

Android fue inicialmente desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró. Android fue presentado en 2007 junto la fundación del "Open Handset Alliance" (un consorcio de compañías de *hardware*, *software* y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el "HTC Dream" y se vendió en octubre de 2008. Desde entonces hasta la actualidad, se han desarrollado diferentes versiones de la plataforma, que han mejorado notablemente sus prestaciones y capacidades. Sin embargo, la diversidad de versiones que conviven en los dispositivos genera algunos problemas en cuanto a compatibilidad entre las aplicaciones móviles existentes.

A fecha de Septiembre de 2015, las diferentes versiones de Android se encuentran distribuidas de la siguiente manera:

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	4.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	3.7%
4.1.x	Jelly Bean	16	12.1%
4.2.x		17	15.2%
4.3		18	4.5%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	15.9%
5.1		22	5.1%

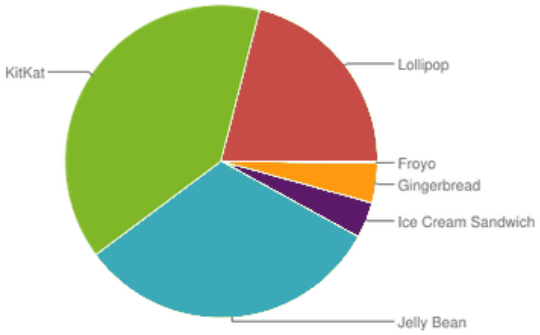

Figura 3. Distribución de las Versiones de Android

En cuanto a su arquitectura (Figura 4), Android contiene una pila de *software* donde se incluye un sistema operativo, *middleware* y aplicaciones básicas para el usuario.

Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores.

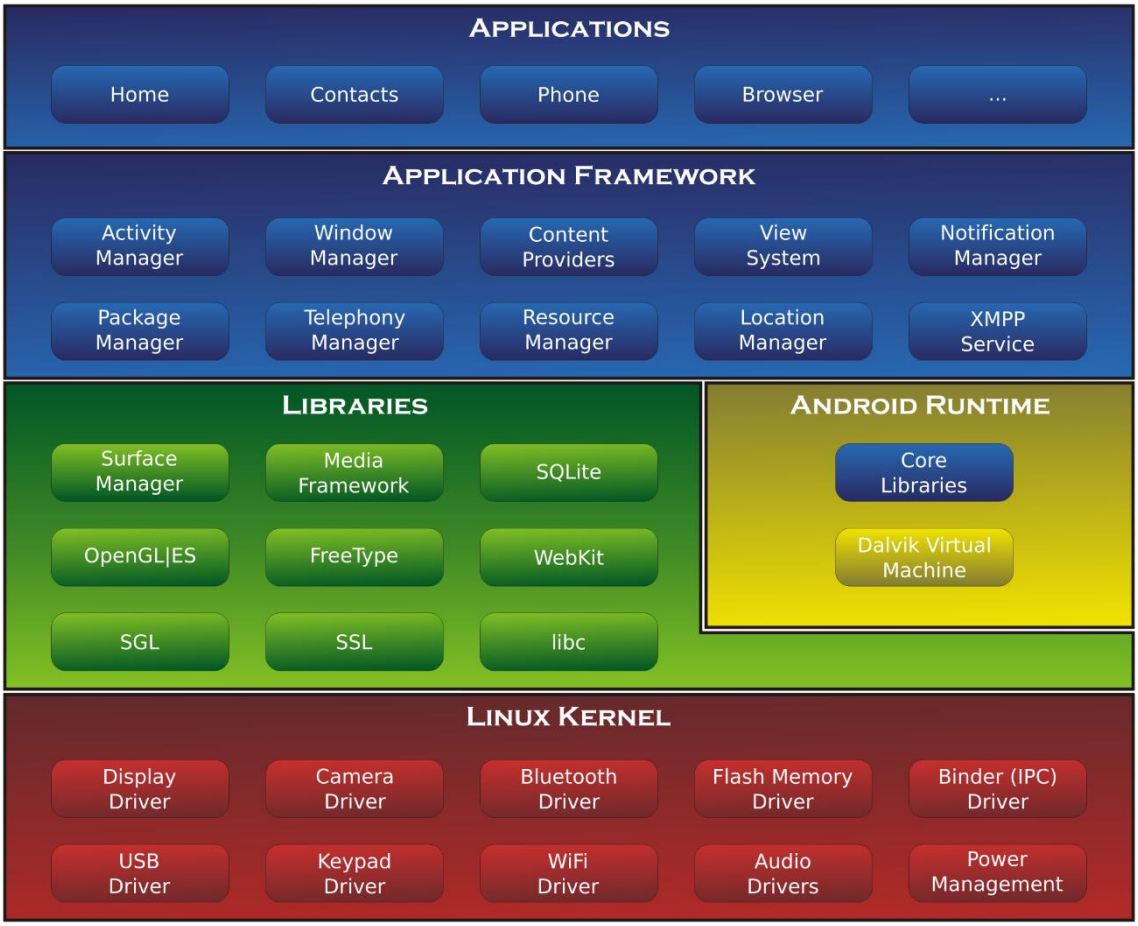

Figura 4. Arquitectura del Sistema Operativo Android

Aplicaciones. Este nivel contiene, tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.

Framework de Aplicaciones. Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo *framework*, representado por este nivel.

Librerías. Esta capa se corresponde con las librerías utilizadas por Android. Éstas han sido escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android.

Tiempo de ejecución de Android. Al mismo nivel que las librerías de Android se sitúa el entorno de ejecución. Éste lo constituyen las "Core Libraries", que son librerías con multitud de clases Java y la máquina virtual Dalvik.

Núcleo Linux. Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el *hardware* disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente *hardware* pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de *hardware*, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro de este núcleo de Linux embebido en el propio Android.

2.2.4 Node.js

Node.js [32] [33] es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Soporta protocolos TCP, DNS y HTTP.

Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables como, por ejemplo, servidores web.

En los servidores tradicionales como Apache, se crea un nuevo hilo por cada conexión cliente-servidor. Esto funciona bien para pocas conexiones, pero crear nuevos hilos es algo costoso, así como los cambios de contexto. Como se puede ver en la Figura 5, a partir de 400 conexiones simultáneas, el número de segundos para atender las peticiones crece considerablemente, pudiendo llegar a saturarse y bloquearse antes muchas conexiones activas; por lo que no es el mejor servidor para lograr máxima concurrencia.

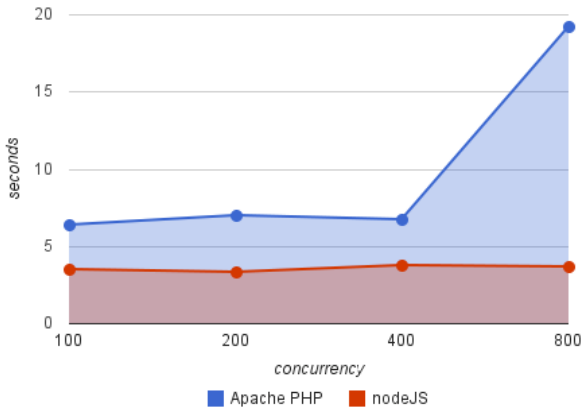

Figura 5. Tiempo de Respuesta en Servidores Apache y NodeJS

Node resuelve este problema cambiando la forma en que se realiza una conexión con el servidor. En lugar de generar un nuevo hilo de para cada conexión (y de asignarle la memoria correspondiente), cada conexión dispara una ejecución de evento dentro del proceso del motor de Node. Si en la aplicación existe una operación bloqueante (I/O por ejemplo), Node creará entonces otro hilo en segundo plano, pero no lo hará sistemáticamente por cada conexión como haría Apache. En teoría, Node puede mantener tantas conexiones como número máximo de archivos descriptores (*sockets*) soportados por el sistema.

Uno de los pocos inconvenientes de Node es que debido a su arquitectura de usar sólo un hilo también que sólo puede usar una CPU. Un método para usar múltiples núcleos sería iniciar múltiples instancias de Node en el servidor y poner un balanceador de carga delante de ellos.

2.2.5 Meteor

Meteor [34] [35] es un *framework* que tiene como objetivo automatizar y simplificar el desarrollo de aplicaciones web reactivas para Node.js que se apoya en bases de datos MongoDB.

Entre sus mayores características, destaca:

JavaScript. Meteor está escrito en JavaScript sobre la base de Node.js, lo que permite utilizar el mismo lenguaje en el lado del cliente y del servidor.

Base de Datos Cliente. Meteor permite decidir qué información de nuestra base de datos queremos mostrar al usuario y se encarga automáticamente de mantener dicha información sincronizada entre el servidor y el cliente. De esta manera, se puede trabajar en el lado del cliente con una copia de los datos y manipularlas sin necesidad de llamar al servidor.

Actualización Reactiva. Cuando se produce un cambio en la base de datos, Meteor se encarga de regenerar solo aquellas partes de la página que se ven afectadas, en lugar de recargar la página completa.

Diseño con Plantillas. La vista de una página se puede definir incluyendo datos dinámicos procedentes de la base de datos. Esto permite la actualización reactiva y nos evita implementar *callbacks* ante cambios en la base de datos.

Protocolo DDP. El DDP ("Distributed Data Protocol") [36] es un protocolo de comunicación bidireccional entre cliente y servidor basado en el *publish/subscribe* [37] que permite la sincronización de la base datos entre clientes y servidor, y llamadas a procedimientos remotos. Al enviar mensajes de tipo JSON a través de WebSockets, la cantidad de información que viaja en cada petición es menor, el tiempo de proceso disminuye y la percepción de rapidez para el usuario final aumenta.

2.2.6 HTML, CSS, JavaScript y jQuery

HTML [38] [39], siglas de "HyperText Markup Language", es el lenguaje de marcado que se emplea para el desarrollo de páginas de internet. Define una estructura básica que está compuesta por una serie de etiquetas que el navegador interpreta y da forma en la pantalla. Es un estándar a cargo del "World Wide Web Consortium" (W3C), organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación.

Por otra parte, las Hojas de Estilo en Cascada o CSS ("Cascading Style Sheets") [40] [41] es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML. El W3C es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

JavaScript [42] es un lenguaje de programación interpretado, dialecto del estándar ECMAScript, que fue diseñado para ser usado dentro de un navegador web. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Actualmente, es uno de los lenguajes

de programación más populares en Internet, y en él se basan tecnologías como las ya mencionadas Node.js o Meteor.

jQuery [43] es una librería JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a las páginas web. jQuery es la biblioteca de JavaScript más utilizada.

3. Diseño

El seguimiento de un paciente consiste en un conjunto de notificaciones que se muestran en el teléfono móvil del mismo. Estas notificaciones le recuerdan la toma de medicación o le formula diversas preguntas para comprobar su estado cada cierto tiempo.

Estas notificaciones se pueden dividir en cuatro grupos, cada uno de los cuales se puede configurar en cualquier momento para que se muestren cada ciertos días y a una hora en concreto del día.

Recordatorio. Se le recuerda al paciente que debe tomarse la medicación. No tiene por qué coincidir con la hora de la toma. Opcionalmente, se le puede preguntar si se tomó la última toma.

Estado de Ánimo. Se le pregunta al paciente como se encuentra en ese momento.

Efectos Secundarios. Consiste en un conjunto de preguntas para comprobar si el paciente experimenta algún efecto secundario. Todas las preguntas son opcionales, pudiéndose mostrar al paciente solamente las que el médico considere adecuadas

Actitud ante la Medicación. Consiste en un conjunto de afirmaciones para comprobar cuál es la aceptación del paciente ante el tratamiento. Igualmente, todas las preguntas son opcionales.

Además de todas estas notificaciones, el paciente podrá tener la oportunidad, en caso de que esté sufriendo una crisis, de mandar un aviso al médico informando de su estado de excepción mediante un botón de “pánico”.

3.1 Diseño General

El sistema se compone de dos actores (médico y paciente) y de tres partes (base de datos, aplicación web y aplicación móvil). El médico tendrá acceso a la aplicación web mientras que el paciente hará uso de la aplicación móvil. Tanto la web como el móvil leerán y escribirán datos en la base de datos.

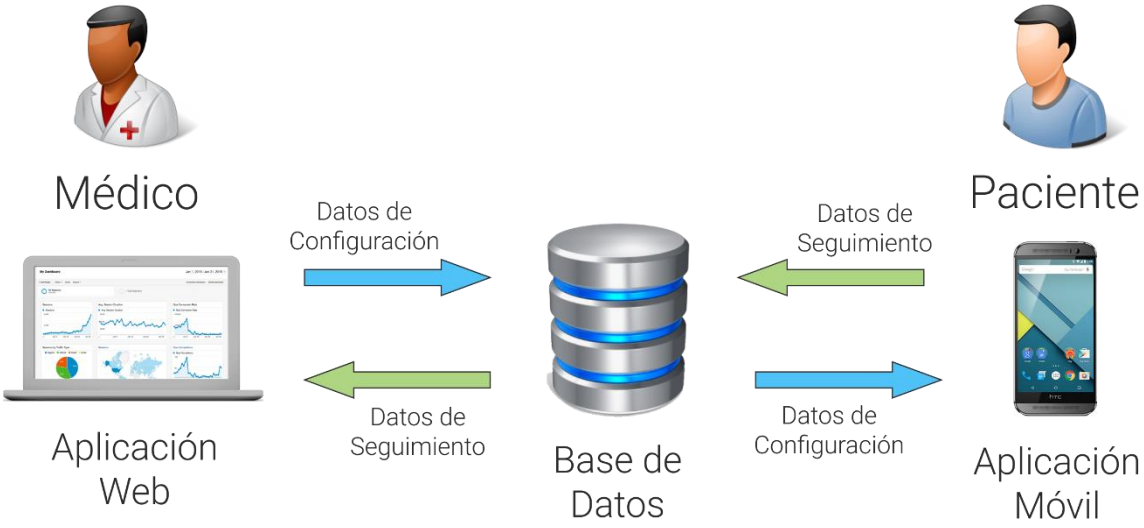

Figura 6. Diseño General del Sistema

Estos datos pueden dividirse de manera lógica en datos de configuración y datos de seguimiento.

Los datos de configuración los establece el médico y consiste en toda la configuración que describe cómo deben comportarse los avisos y notificaciones en el dispositivo móvil del paciente. Estos datos se introducen a través de la aplicación web a la que solo tendrá acceso el médico.

Los datos de configuración serán recibidos por la aplicación móvil del paciente, la cual se programará para mostrar las notificaciones de acuerdo a la configuración. Estos datos de configuración podrán cambiar en cualquier momento, por lo que la aplicación móvil deberá adaptarse a estos cambios.

Cuando una notificación se muestre y pida *feedback* al paciente (como por ejemplo, el estado de ánimo), se generará el otro tipo de datos, datos de seguimiento, los cuales serán enviados a la base de datos.

Cuando el médico acceda a la aplicación web, podrá visualizar todos los datos a través de diversas tablas y gráficas. Esta deberá actualizarse automáticamente cuando nuevos datos de seguimiento se inserten en la base de datos.

Pasemos a ver ahora con más detalle cada una de las tres partes principales del sistema.

3.2 Diseño de la Base de Datos

Los datos son almacenados en una base de datos MongoDB, de tipo NoSQL. Debido a sus características, ya explicadas en los capítulos 2.2.1 y 2.2.2, resulta idónea para nuestro proyecto.

Cada paciente tiene su documento JSON que describe su estado, la configuración de sus notificaciones y sus progresos. Exactamente el documento contiene los siguientes campos:

_id : String

Es el identificador interno que proporciona MongoDB al registro.

startDate :ISODate

Indica el día que el paciente empezó su seguimiento.

settings :Object

Describe la configuración sobre cómo deben comportarse los avisos y notificaciones en el móvil del paciente. Este objeto se divide en los cuatro tipos de notificaciones: recordatorio (`reminder`), estado de ánimo (`feeling`), efectos secundarios (`sideEffects`) y actitud ante la medicación (`attitude`):

Para cada uno de ellos se especifica si está activado (`enabled`). En caso positivo, se indica la periodicidad (en días) y la hora a la que se muestra la notificación.

En el caso concreto del recordatorio, además se incluye si deseamos solamente recordar al paciente la toma de la medicación o también preguntarle si se tomó la anterior (`confirmation`).

Para los casos de Efectos Secundarios y Actitud, se pueden seleccionar qué preguntas concretas se le formularán al paciente.

Un ejemplo completo de configuración podría ser el mostrado por el Código 1.

```
settings: {
    reminder: {
        enabled: true,
        periodicity: 1,
        hour: 12,
        confirmation: true
    },
    feeling: {
        enabled: true,
        periodicity: 2,
        hour: 18
    },
    sideEffects: {
        enabled: true,
        periodicity: 5,
        hour: 20,
        digestion: false,
        movement: true,
        appetite: true,
        sex: false,
        sleep: false
    },
    attitude: {
        enabled: false,
        periodicity: 5,
        hour: 20,
        relaxation: true,
        thoughts: false,
        unnatural: true,
        prevention: false
    }
}
```

Código 1. Ejemplo de Configuración

En este ejemplo, vemos que se ha configurado un recordatorio de medicación con confirmación diaria a las 12h, una consulta de estado de ánimo cada dos días a las 18h y una consulta de efectos secundarios cada 5 días a las 20h que consta de las preguntas relacionadas con el movimiento y el apetito. La consulta de actitud permanece desactivada.

panics : Array[ISODate]

Almacena en un vector todos los momentos en los que el paciente ha considerado que se encontraba en un estado de excepción y ha pulsado el botón de pánico.

records :Object

Este objeto registra todos los datos de seguimiento generados por el paciente. Tiene tantos campos como posibles preguntas se le puedan hacer al paciente y cada uno de ellos es un vector con otro objeto que indica el día que se ha generado el dato (`day`), y el propio dato codificado en un número (`value`).

El Código 2 muestra un posible registro de los primeros 6 días de tratamiento de un paciente (obedece a la configuración vista en el Código 1).

```
records: {
    reminder: [
        {day: 1, value: 1},
        {day: 2, value: 1},
        {day: 3, value: 1},
        {day: 4, value: -1},
        {day: 5, value: 1},
        {day: 6, value: 0}
    ],
    feeling: [
        {day: 2, value: 3},
        {day: 4, value: -1},
        {day: 6, value: 5},
    ],
    digestion: [],
    movement: [
        {day: 5, value: 0}
    ],
    appetite: [
        {day: 5, value: 2}
    ],
    sex: [],
    sleep: [],
    relaxation: [],
    thoughts: [],
    unnatural: [],
    prevention: []
}
```

Código 2. Ejemplo de Registro de Seguimiento

3.3 Diseño de la Aplicación Móvil

3.3.1 Definición de Tareas

La aplicación móvil debe ser capaz de realizar las siguientes tareas:

- Obtener la configuración de las notificaciones de la base de datos
- Programar sus notificaciones para ser mostradas de acuerdo a la configuración
- Mostrar notificaciones que, según el caso, abra la aplicación para poder realizar el cuestionario
- Enviar los datos recopilados a la base de datos
- Ser capaz de detectar cambios en la configuración y reprogramar las notificaciones
- Volver a programar las notificaciones cuando el sistema operativo se inicie
- Ser capaz de hacer todo lo anterior manteniendo la aplicación cerrada
- Mostrar al paciente un botón de pánico para avisar a su médico
- Mostrar cuando se producirán las próximas notificaciones

3.3.2 Conceptos Previos

Para explicar el diseño llevado a cabo para realizar estas tareas, primero debemos conocer algunos conceptos dentro del entorno de la programación Android. Concretamente:

- Actividades
- Servicios
- Notificaciones
- *Intents*
- *BroadcastReceivers*
- Alarmas

Actividades

Las actividades es el componente principal de cualquier aplicación Android, que proporciona una pantalla con la que el usuario puede interactuar con el fin de realizar algo. Esta pantalla incorpora elementos de interfaz gráfica, como texto, imágenes, botones, vídeos, campos de texto, *checkboxes*, etc. Una aplicación puede contener diversas actividades, y estás pueden relacionarse entre sí, enviándose datos o generando una secuencia de actividades para completar un proceso.

Ejemplo de actividades las podemos encontrar en casi la totalidad de aplicaciones existentes, por ejemplo, Gmail, YouTube, WhatsApp, etc.

Servicios

Los servicios son componentes que ejecutan operaciones en segundo plano sin la presencia de una interfaz gráfica. Sirven principalmente para realizar tareas de manera repetitiva y potencialmente de larga duración como la descarga de un fichero de Internet, el procesamiento de datos o la carga de contenidos a un servidor. Pueden ser invocados desde diferentes partes de la aplicación y su ejecución continúa incluso si el usuario cambia de aplicación, al contrario que sucede con las actividades.

Notificaciones

Las notificaciones son mensajes que aparecen en el área de notificación, situada en la parte superior de la pantalla. Cuando una notificación aparece por primera vez muestra un icono, un pequeño mensaje y opcionalmente, un sonido o vibración.

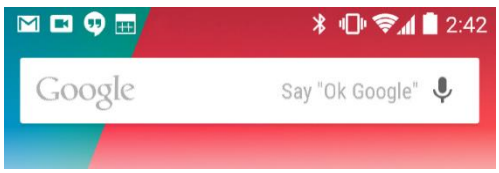

Figura 7. Área de Notificación con 4 notificaciones

Desplegando el área de notificación se puede visualizar la notificación completa, que consiste en un icono de mayor tamaño, un título y una breve descripción. Opcionalmente, pueden incorporarse botones para interactuar con la notificación sin tener que abrir una nueva actividad.

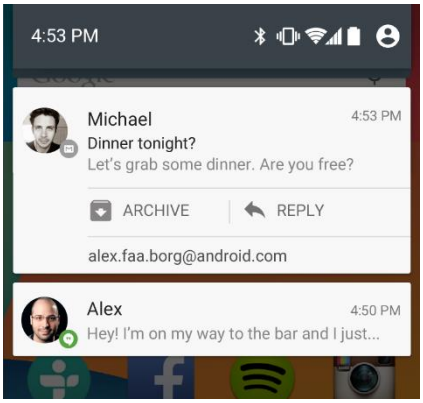

Figura 8. Área de Notificación Desplegada

Intents

Un *Intent* es una descripción abstracta de una operación a realizar. Sirve para comunicar al sistema operativo que tenemos la intención de realizar una determinada acción. Esa acción puede ser una acción que Android conoce y sabe cómo gestionar, o puede ser una acción creada por el desarrollador que solo funcione en el ámbito de su aplicación.

Por ejemplo, puede que una aplicación de edición de video tenga como característica compartir el video con otras personas o redes sociales. En lugar de implementar el envío de videos en la aplicación, la filosofía Android facilita que se apoye en otras aplicaciones ya instaladas en el dispositivo móvil para realizar esa tarea en concreto. Para ello, la aplicación puede enviar un *Intent* cuya acción sea `ACTION_SEND`. Android conoce qué otras aplicaciones son capaces de responder a este *Intent* y muestra una lista para que el usuario elija la que desee.

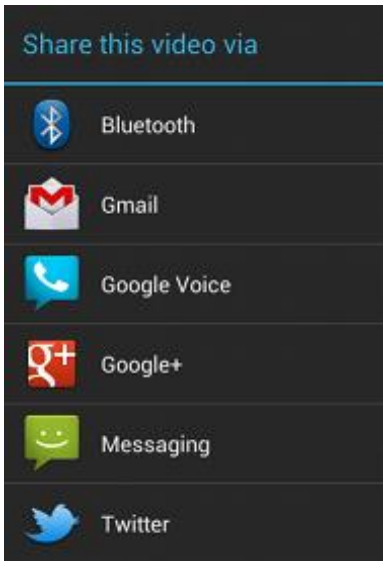

Figura 9. Listado de Aplicaciones que responden a ACTION_SEND

De igual manera, dentro de una misma aplicación, se pueden enviar diferentes *Intents* que describan una acción de creación propia para comunicar ciertas partes de una misma aplicación.

Broadcast Receivers

Para que una aplicación pueda recibir y procesar un *Intent* de una acción en concreto debe registrarlo previamente ante el sistema operativo. De esta manera, el SO sabrá que la aplicación es capaz de responder a esa acción, e invocará una parte de nuestro código que reciba el *Intent* para que lo procese. A esta pieza de código se le llama el *BroadcastReceiver*. Si se desea que una aplicación sea capaz de procesar varias acciones diferentes, deberá tener varios *BroadcastReceivers* diferentes. Cada uno de ellos se encargará de procesar una acción en concreto.

También se puede ver el envío de los *Intents* como una manera de generar eventos. De esta manera podemos destacar uno en concreto, el que genera Android cuando se inicia el sistema operativo: `android.intent.action.BOOT_COMPLETED`.

Esta acción es generada cuando el sistema operativo ha terminado de iniciarse y resulta especialmente útil para realizar ciertas tareas lo más pronto posible sin tener que iniciar manualmente la aplicación. Gmail, por ejemplo, requiere comprobar la existencia de correo nuevo sin esperar a que el usuario entre en la aplicación, por lo que, tal y como hemos comentado, Gmail tendría registrado un *BroadcastReceiver* para recibir y procesar el evento `BOOT_COMPLETED`.

Alarmas

Las alarmas son mecanismos del sistema Android que permiten realizar operaciones programadas en cierto momento y/o cada cierto intervalo fuera del ciclo de la aplicación que lo establece. Por ejemplo, se puede crear una alarma para que un servicio de previsión meteorológica descargue información actualizada una vez al día. De esta manera, aunque la aplicación se cierre, la alarma seguirá existiendo y el servicio se lanzará a la hora configurada, incluso si el dispositivo móvil se encuentra en estado de reposo. Esto ayuda a minimizar la demanda de memoria, ya que ni la aplicación estará en memoria ni será necesario crear hilos adicionales para esperar un día entero.

3.3.3 Diseño de la Aplicación Móvil

En primer lugar, debemos pensar cuantas pantallas, o actividades, queremos presentar al usuario. Al haber 4 grupos de notificaciones (recordatorio, estado de ánimo, efectos secundarios, actitud ante la medicación), sería lógico pensar en 4 actividades diferentes. Sin embargo, vamos a resolver el recordatorio de toma de medicación en una respuesta de tipo Sí/No, que resulta más cómodo e inmediato de incluir dentro de la propia notificación, sin tener que crear ninguna actividad adicional para ello.

Además, necesitamos una actividad principal para mostrar un panel informativo con la hora de las próximas notificaciones y el botón de pánico.

De esta forma, nuestro diseño comienza con 4 actividades:

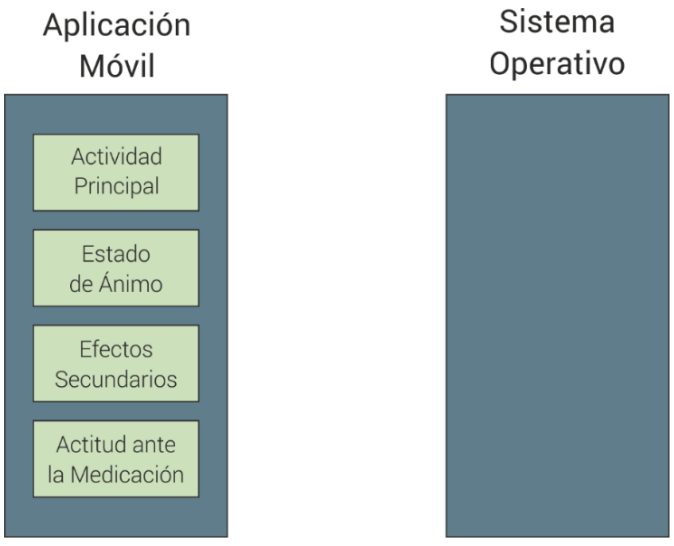

Figura 10. Diagrama con Actividades

Actividad Principal es la actividad que se muestra al abrir la aplicación. El resto se mostrarán a través de notificaciones y no podrán ser mostradas a voluntad del usuario.

Cuando se ejecuta por primera vez la actividad principal, debe, en primer lugar, descargar la configuración de las notificaciones ubicada en la base de datos (la estructura "`settings`" vista en el apartado 3.2). Lo hará a través de los mecanismos que tiene Android para conectarse a Internet. Una vez recibido, se crearán 4 alarmas y serán configuradas para que disparen un evento (un *Intent* con una acción que identifique el grupo de notificación) a la hora y con la frecuencia determinada por la configuración. Al ser las alarmas un elemento del sistema operativo y no de nuestra aplicación, nos aseguramos que serán disparadas aunque nuestra aplicación haya sido destruida por completo. También configuraremos una alarma más para que consulte si ha habido cambios en la configuración, con frecuencia diaria.

En este estado, el diagrama de la aplicación es el siguiente:

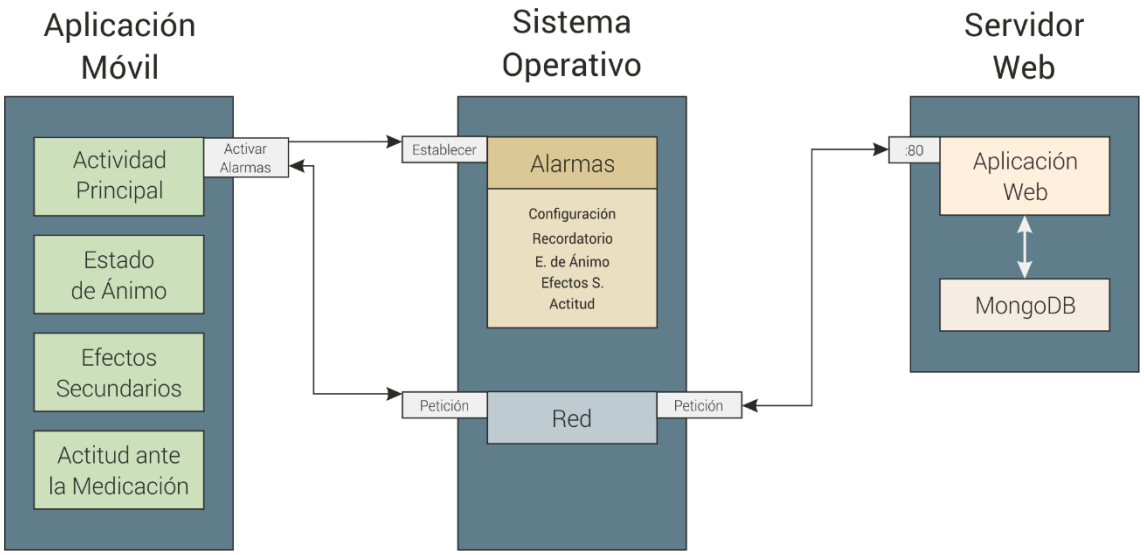

Figura 11. Diagrama con Alarmas

Cuando llega el momento a la que una alarma ha sido establecida, se genera un evento que debe ser capturado por nuestra aplicación para procesarla. Para ello, utilizamos los *Broadcast Receivers*.

Cada *Broadcast Receiver* recibe un tipo de evento, por lo que necesitamos 5 de ellos. Cada uno de ellos se encargará de procesar cada evento de manera independiente.

Si el evento capturado es el de configuración, la aplicación vuelve a consultar la configuración vía web para comprobar si ha habido cambios. En caso afirmativo, cancela el resto de alarmas y las vuelve a crear con la nueva configuración. También actualizaría el panel de próximas notificaciones de la Actividad Principal si esta estuviera mostrándose al usuario.

Si, por el contrario, el evento capturado es el asociado a una de las preguntas que se le debe hacer al paciente, se mostrará una nueva notificación en el área de notificaciones del sistema operativo para que el usuario se percate de que debe responderla.

El sistema de notificaciones es otro elemento del sistema operativo y no de nuestra aplicación, por lo que no es necesario tener la aplicación abierta para que estas notificaciones se muestren.

A excepción de la notificación del recordatorio (que mostrará los botones de respuesta en la propia notificación), el resto de notificaciones abrirá una actividad para que el paciente pueda responder a las preguntas programadas.

Hay que tener en cuenta que todas las alarmas que se establezcan durante el ciclo de vida de la aplicación serán suprimidas cuando el sistema operativo se cierre, por lo que tendremos que volver a establecerlas de nuevo al iniciarse.

Para ello, capturamos el evento que genera el sistema operativo cuando termina de iniciarse, `android.intent.action.BOOT_COMPLETED`. Lo hacemos mediante un nuevo *Broadcast Receiver*, que inicia de nuevo todas las alarmas.

El esquema a continuación muestra el estado actual del diseño:

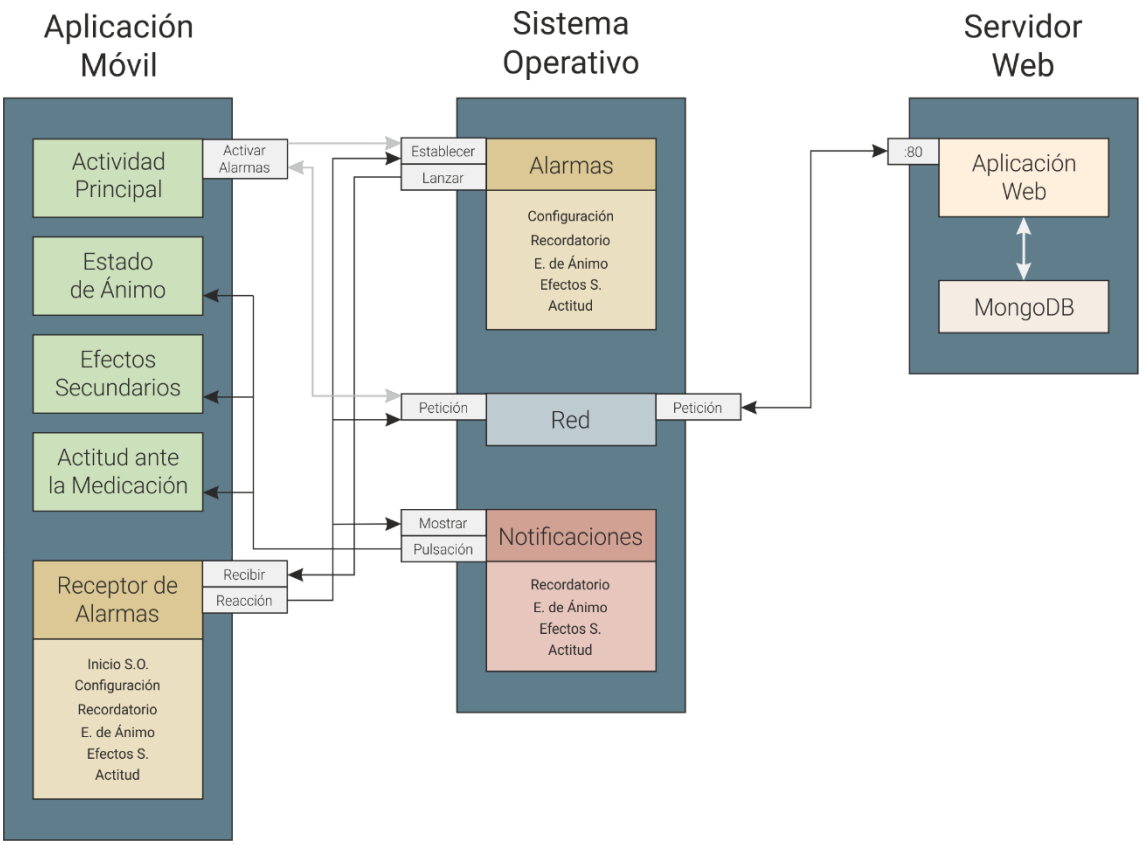

Figura 12. Diagrama con Broadcast Receivers y Notificaciones

Cada vez que se muestra una notificación y se abre su correspondiente actividad formulando una pregunta al usuario, este genera un dato de seguimiento al contestar a dicha pregutna.

Ese dato de seguimiento debe almacenarse en la base de datos de nuestro servidor. Debido a que el *framework* empleado para la creación de la aplicación web es Meteor (cuyas bondades y características detallaremos en la siguiente sección), resulta especialmente fácil y seguro que la inserción de datos en la base de datos se realice a través de la aplicación web y no directamente en la base de datos. Por lo que para insertar el dato solo será necesario acceder al servidor con una ruta concreta que defina la operación. El servidor web será el encargado de *parsear* la URL,

realizar las comprobaciones necesarias, insertar el dato de la base de datos y devolver un ACK al dispositivo móvil para confirmar que todo ha salido correctamente.

De igual manera, el botón de pánico ubicado en la Actividad Principal también inserta un dato en la base de datos del servidor.

Por tanto, el diagrama completo de la aplicación móvil tendría el siguiente aspecto:

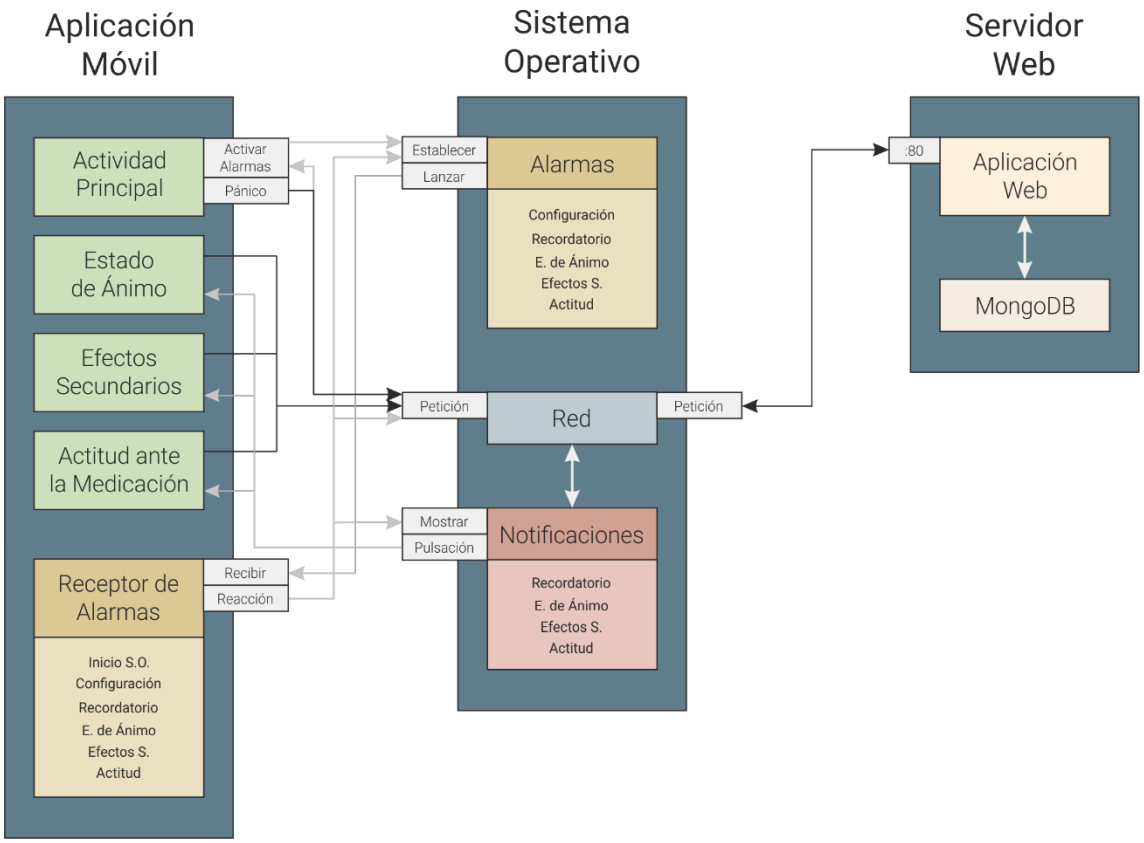

Figura 13. Diagrama Completo

3.4 Diseño de la Aplicación Web

3.4.1 Definición de Tareas

Las tareas que debe ser capaz de realizar la aplicación web son:

- Obtener los datos de configuración y seguimiento de la base de datos
- Modificar los datos de configuración
- Autoactualizarse cuando la base de datos reciba datos nuevos de seguimiento
- Representar los datos de una manera cómoda e intuitiva
- Servir de interfaz para consultar e insertar datos en la base de datos

3.4.2 Conceptos Previos

Para explicar el diseño llevado a cabo para realizar estas tareas, primero debemos conocer algunos conceptos dentro del *framework* Meteor. Concretamente:

- Separación en Cliente y Servidor
- Funciones en Servidor
- Enrutamiento
- Plantillas
- Protocolo DDP

Separación en Cliente y Servidor

Meteor permite dividir la aplicación web en una parte cliente y otra parte servidor. La parte cliente serán todos aquellos ficheros (.html, .css, .js, imágenes, etc) que se enviarán al cliente y se ejecutarán en su máquina local. Al estar en local, nada le impedirá cambiar parte del código para modificar su funcionamiento o sortear su seguridad. Para evitarlo, parte del código puede residir en el servidor y no ser enviado al cliente, evitando así que sea leído o modificado, incrementando la seguridad del sistema. También, si el sistema exige la ejecución de cálculos complejos para una máquina cliente común, se pueden ejecutar en la parte servidor sin sobrecargar el cliente, o incluso distribuirlo en un clúster.

Por ejemplo, supongamos que al insertar un dato a la base de datos, se comprueba que su valor no sea superior a un límite dado. Si esa comprobación se realiza en el código cliente, será fácil modificarlo para permitir insertar cualquier dato. Sin embargo, incluir el código de comprobación en el servidor solucionaría este problema.

Por tanto, a la hora de diseñar una aplicación web con Meteor, es necesario pensar que partes de nuestra aplicación son sensibles e incluirlas en la parte servidor.

Funciones en Servidor

Ampliando un poco más el anterior apartado, conviene detallar que cada pieza de código que se incluye en el servidor, se hace en forma de funciones, por lo que podrán ser ejecutadas desde el cliente y desde el *router* (explicado a continuación).

Enrutamiento

El enrutamiento es el sistema por el cual una URL es *parseada* e interpretada, generando una acción en concreto, ya sea ir a una sección de la web, hacer una consulta a una base de datos u obtener el resultado de un cálculo.

Al igual que el resto de la aplicación, el código de enrutamiento puede residir en el servidor y ser enviado al cliente.

Por ejemplo, se puede programar el *router* para que las URL de tipo "`/post/<type>/<value>`" inserte en la base de datos el documento "`{type: value}`", donde `type` y `string` sean los valores introducidos en la URL. Adicionalmente, se podría comprobar que `value` sea un número y que resida dentro de un rango concreto. En ese caso, sería recomendable situar este código en el lado de servidor para evitar modificaciones.

Plantillas

Las plantillas permiten describir funcionalmente en un fichero HTML el contenido dinámico proveniente de una base de datos y mantenerlas constantemente actualizadas. De esta manera, no habría que escribir código Javascript que detectase cambios en la base de datos y modificase el DOM ya que ese comportamiento ya está incluido en el *framework*. Esta descripción funcional resulta mucho más fácil, compacta y comprensiva.

Por ejemplo, si deseamos hacer una lista con los campos de nombre y edad de una lista de personas de una base de datos que puede cambiar continuamente, sería suficiente con:

```
<div>
  <ul>
    {{#each people}}
        <li>Nombre:{{name}} Edad:{{age}}</li>
    {{/each}}
  </ul>
</div>
```

Código 3. Ejemplo de Plantilla en Meteor

Siendo `people` el array de documentos, y siendo `name` y `age` dos campos de cada documento.

De esta manera, si en algún momento, se modifica en la base de datos alguno de los campos o si se añade o elimina un nuevo documento, se actualizarían los datos automáticamente sin necesidad de código Javascript adicional.

Esto es posible ya que cliente y servidor se comunican mediante el protocolo DDP.

Protocolo DDP

DDP ("Distributed Data Protocol") [36] es un protocolo de comunicación bidireccional para recibir datos estructurados desde un servidor, a la vez que recibir actualizaciones cuando los datos cambian.

DDP se puede definir como "REST con WebSockets". Al igual que REST, es un método sencillo y pragmático que proporciona una API de comunicación. Pero DDP al estar basado en WebSockets, permite ser utilizado para entregar actualizaciones en vivo tan pronto como se produzcan cambios en los datos. Su implementación se basa en el intercambio de mensajes JSON.

3.4.3 Diseño de la Aplicación Web

La aplicación web no solo sirve para establecer las configuraciones de las notificaciones y visualizar los datos de seguimiento en un navegador, sino también como interfaz para acceder a la base de datos desde la aplicación móvil (tanto para consultar como para modificar), añadiendo así una capa de seguridad y comprobación de datos antes de acceder a ella.

La forma de acceder a la base de datos es accediendo a una URL que el *router parseará*, validará y ejecutará adecuadamente, devolviendo finalmente un valor de tipo JSON, ya sea como resultado de una consulta o como confirmación de una modificación.

Como hemos visto en anteriores apartados, la aplicación móvil necesita acceder a la base de datos para realizar dos tareas: obtener la configuración actual e insertar datos de seguimiento.

Para la primera tarea, enviaría una petición a la URL "`/getSettings`". La aplicación web recibiría la petición, haría la consulta correspondiente a la base de datos y devolvería un JSON con dicha configuración.

Para enviar los datos de seguimiento, enviaría una petición a la URL "`/post/<type>/<value>`", donde `type` es un `token` que define el tipo de datos de seguimiento (recordatorio, estado de ánimo, etc) y `value` es el valor numérico del dato de seguimiento. De esta manera, la aplicación web, al recibir la petición, *parsearía* la URL y realizaría la inserción en la base de datos de acuerdo a los valores `type` y `value`.

Por otro lado, atendiendo al navegador web, solo debemos diseñar la interacción al realizar la petición de envío de la página web, la cual se conseigue a través de su URL raíz. El resto de interacciones, todas aquellas involucradas en mantener la vista sincronizada con los datos de la base de datos, es responsabilidad de las múltiples librerías del *framework* Meteor, y resulta transparente para nuestro diseño.

Por tanto, el diagrama de interacciones quedaría de la siguiente manera:

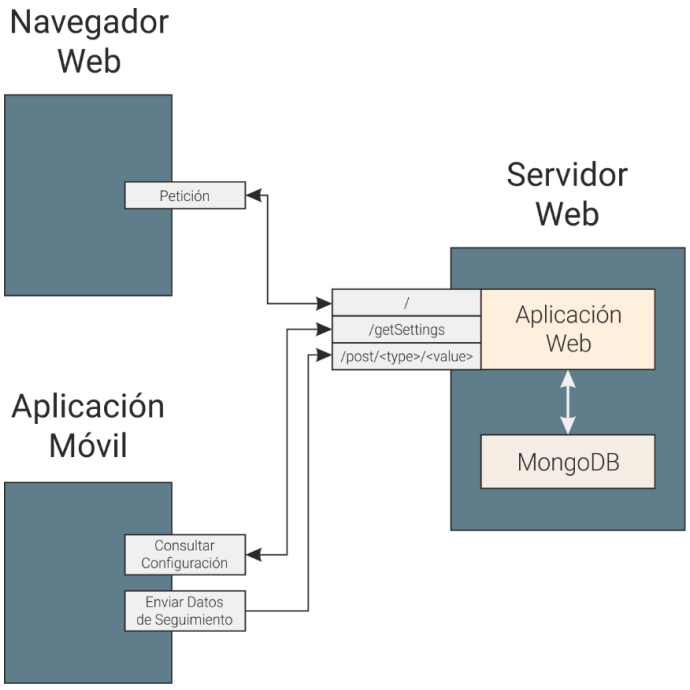

Figura 14. Diagrama de la Aplicación Web

Una petición al directorio raíz responde con el contenido de la página web. Aquí es donde se conectará el médico para comprobar los progresos del paciente. En ella se pueden diferenciar 4 secciones diferentes:

Datos del Paciente. Donde se muestran los datos personales básicos y la fecha de inicio del tratamiento

Avisos de Pánico. En esta sección se puede comprobar en qué momentos el paciente presionó el botón de pánico.

Configuración. Aquí el medico puede modificar los parámetros que definen cómo deben comportarse los avisos y notificaciones en el dispositivo móvil del paciente.

Registros. En esta parte, se muestran diversas representaciones gráficas de cada uno de los datos de seguimiento, desde el inicio del tratamiento hasta la actualidad.

4. Implementación de un Prototipo

4.1 Entorno de Desarrollo

Para la implementación del proyecto, se han utilizado un conjunto de herramientas y aplicaciones que pasamos a detallar a continuación.

4.1.1 Android Studio

- Versión 1.3
- Licencia Apache 2.0
  http://www.apache.org/licenses/LICENSE-2.0

Android Studio [44] es el IDE ("Integrated Development Environment") oficial para el desarrollo de aplicaciones móviles para Android, basado en el IntelliJ IDEA, otro IDE de desarrollo para aplicaciones Java. Desde la primera versión estable de Android Studio en Diciembre de 2014, ha sustituido a "Eclipse Android Development Tools" como el entorno de desarrollo oficial por Google.

Entre sus mayores características podemos destacar:

- Editor WYSIWYG con *renderización* en tiempo real.
- Soporte para construcción basada en Gradle.
- Consola de desarrollado con: consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Refactorización de código y arreglos rápidos.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad y compatibilidad de versiones.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para Android Wear.
- Emulador para probar y testear aplicaciones.

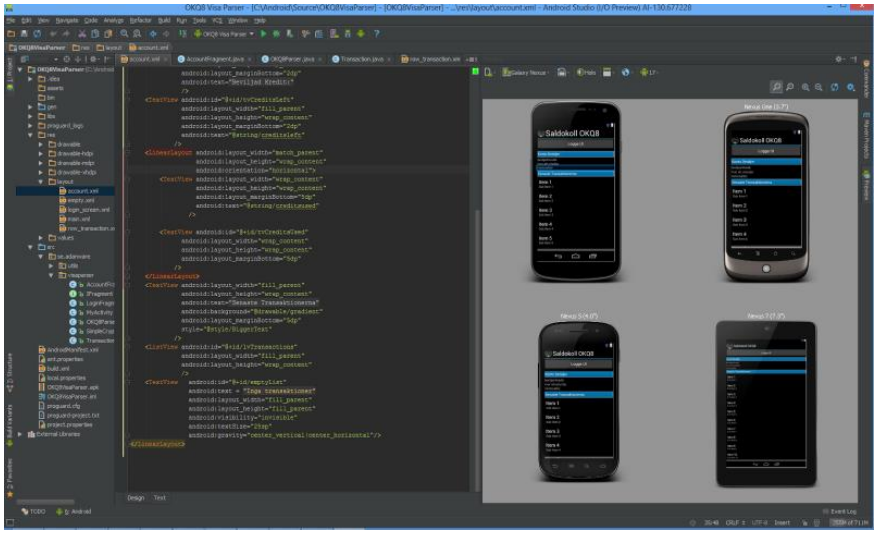

Figura 15. Android Studio

Android Studio será la herramienta empleada para implementar la aplicación móvil de nuestro proyecto.

4.1.2 Genymotion

- Versión 2.5.2
- Licencia privada con versión personal gratuita https://www.genymotion.com/#!/legal/terms-and-conditions

Genymotion [45] es un emulador Android que sirve para testear aplicaciones. Entre otras caracterices, permite:

- Virtualización de la CPU
- Aceleración OpenGL
- Emulación de sensores: GPS, micrófono, cámara, batería, *multi-touch*, acelerómetro
- Integración con Android Studio
- Herramienta de línea de comandos
- Posibilidad de *screenshots* y *screencasts*.
- Versiones compatibles desde la 2.3 hasta la 5.1

A pesar de que Android Studio dispone de su propio emulador, este resulta lento y pesado. Genymotion es más rápido, ligero y dispone de ciertas facilidades que el emulador de Android Studio no posee; por lo que ha sido Genymotion el emulador con el que se han hecho las pruebas y comprobaciones de nuestra aplicación móvil durante el desarrollo.

Figura 16. Genymotion

4.1.3 Robomongo

- Versión 0.8.4
- Licencia: GPL http://www.gnu.org/licenses/gpl-3.0.en.html

Robomongo [46] es una herramienta multiplataforma de gestión de bases de datos Mongo DB, que incorpora el mismo motor JavaScript que utiliza el *shell* de MongoDB 2.2. Robomongo incluye una interfaz gráfica con diversas ventajas como el resaltado de sintaxis, autocompletado, diferentes modos de visualización de la base de datos (texto, árbol, personalizados), múltiples conexiones simultáneas, etc. Robomongo nos servirá para poder consultar en todo momento el estado de la base de datos y realizar modificaciones sin pasar por la aplicación web o móvil, a modo de realización de pruebas y comprobaciones durante la fase de desarrollo.

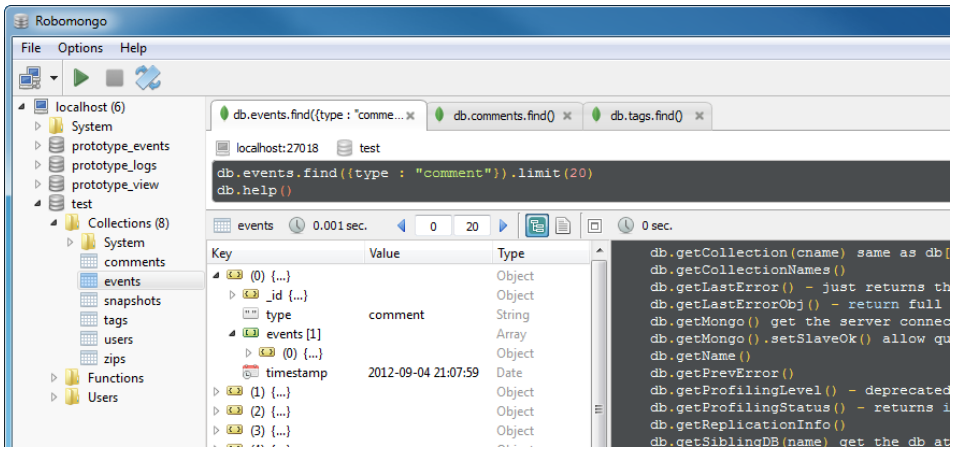

Figura 17. Robomongo

4.1.4 Navegadores Web

Para comprobar el correcto funcionamiento de nuestra aplicación web, empleamos 5 diferentes navegadores web, que abarcan el 95% del mercado actual:

- Google Chrome 45 [47]
- Mozilla Firefox 40 [48]
- Microsoft Internet Explorer 11 [49]
- Opera 31 [50]
- Maxthon 4.4.6 [51]

4.2 Implementación del Prototipo

En esta sección vamos a mostrar el resultado de la implementación de la aplicación móvil y web.

4.2.1 Aplicación Móvil

La actividad principal de la aplicación móvil es mostrada cuando se abre la aplicación y es la única que puede ser abierta voluntariamente por el usuario. Incluye, como se puede ver en la Figura 18, un interruptor para activar o desactivar todo el sistema, un panel con la fecha y hora de las próximas notificaciones y un botón de pánico para avisar al médico de un estado de excepción.

Una vez activada la aplicación, el sistema de notificaciones es independiente por lo que la actividad puede ser destruida sin afectar a sus funcionalidades.

Figura 18. AppMóvil - Pantalla Principal

Cuando una de las alarmas programadas es lanzada, se muestra una notificación y el dispositivo genera una vibración. Como hemos explicado, existen 4 tipos de notificaciones (recordatorio, estado de ánimo, efectos secundarios y actitud ante la medicación). Solo la notificación de recordatorio puede incluir los botones de respuesta en la propia notificación. El resto, enlaza a una nueva actividad al ser pulsada. Las Figuras 19, 20, 21 y 22 muestran los 4 tipos de notificación.

Figura 19. AppMóvil - Notificación de Recordatorio

Figura 20. AppMóvil - Notificación de Estado de Ánimo

Figura 21. AppMóvil - Notificación de Efectos Secundarios

Figura 22. AppMóvil - Notificación de Actitud

La notificación de estado de ánimo abre una nueva actividad que le pregunta al paciente como se encuentra en ese momento. Las respuestas posibles son:

- Muy bien
- Bien
- Un poco bien
- Normal
- Un poco mal
- Mal
- Muy mal

Al pulsar una respuesta, se envía el dato de seguimiento al servidor y la actividad de cierra.

Figura 23. AppMóvil - Pantalla Estado de Ánimo

La notificación de efectos secundarios abre otra actividad donde formula diversas preguntas para comprobar si el paciente experimenta algún efecto secundario. Las preguntas son:

- ¿Cómo te ha sentado la medicación?
- ¿Has tenido alteraciones en tus movimientos, rigidez o inquietud?
- ¿Has tenido variación en tu apetito?
- ¿Problemas en tu actividad sexual?
- ¿Has tenido alteraciones en el sueño?

Todas las preguntas son opcionales, pudiéndose mostrar al paciente solamente las que el médico considere adecuadas.

Una vez el paciente pulsa el botón de finalización, se envía los datos al servidor y la actividad se cierra.

En la Figura 24 que representa la actividad se han mostrado las 4 primeras preguntas.

Figura 24. AppMóvil - Pantalla Efectos Secundarios

La notificación de actitud abre otra actividad donde se presenta un conjunto de afirmaciones para comprobar cuál es la aceptación del paciente ante el tratamiento. Las afirmaciones son:

- La medicación me relaja
- Mis pensamientos son más claros con la medicación
- Es antinatural para mi mente y mi cuerpo estar controlado por medicamentos
- Por estar con medicación puedo prevenir caer enfermo
- Las decisión de qué preguntas incluir y a qué hora mostrarlas es tomada por el médico

Y las posibles contestaciones para cada una de las afirmaciones son:

- Totalmente en desacuerdo
- En desacuerdo
- Ni de acuerdo ni en desacuerdo
- En acuerdo
- Totalmente de acuerdo

Todas las preguntas son opcionales.

Figura 25. AppMovil - Pantalla Actitud ante la Medicación

Una vez el paciente pulsa el botón de finalización, se envía los datos al servidor y la actividad se cierra.

En la Figura 25 se ha mostrado solo la segunda pregunta.

4.2.2 Aplicación Web

La aplicación web, junto al servidor Node.js y la base de batos MongoDB, se encuentran instalados en una máquina virtual con procesador de 2GHz y 1GB de RAM, con sistema operativo Ubuntu 14.04.2 LTS

La plataforma Cloud tiene 224 *cores* y 512 GB de RAM, con una "Storage Area Network" (SAN) de 10 TB. Cada nodo físico es un biprocesador Xeon E5-2683v2 a 2.0GHz, con 28 *cores* en total, y dispone de 64GB de RAM. La plataforma Cloud utiliza OpenNebula 4.8.

La aplicación web muestra 4 secciones diferentes:

Datos del paciente. Al tratarse actualmente de un prototipo, la aplicación no incluye datos personales del paciente, solo la fecha de inicio del tratamiento y el día de tratamiento actual.

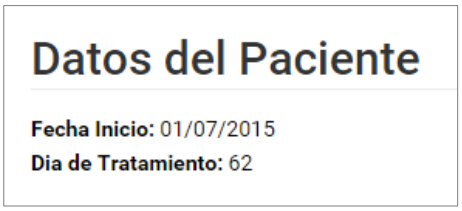

Figura 26. AppWeb - Datos del Paciente

Avisos de Pánico. Muestra, si los hay, los avisos de pánico que ha generado el paciente, detallando la fecha y la hora de la pulsación.

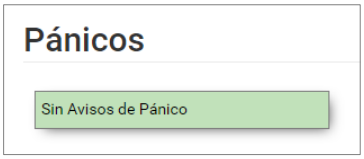

Figura 27. AppWeb - Sin Avisos de Pánico

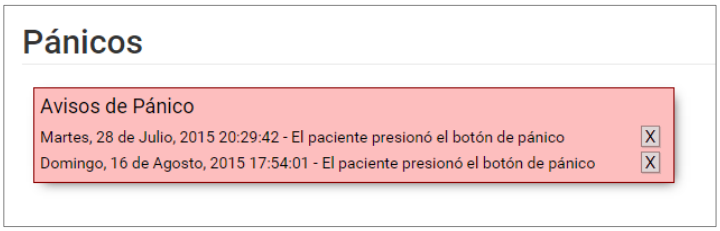

Figura 28. AppWeb - Avisos de Pánico

Gracias a las librerías de Meteor que proporcionan una sincronización automática entre cliente y servidor, tan pronto como se genere un aviso, el cliente será actualizado sin tener que recargar la página.

Configuración. En esta sección el médico puede decidir qué tipo de preguntas realizar el paciente, con qué frecuencia y a qué hora del día. Esta configuración puede ser modificada en cualquier momento del tratamiento, ya que la aplicación móvil está programada para detectar este tipo de cambios y reprogramar todas sus alertas.

En la Figura 29 se representa la siguiente configuración:

Todos los días a las 9 de la mañana se le recuerda al paciente la toma de la medicación. No se le pregunta si tomó la anterior.

Cada 2 días a las 2 de la tarde, se le pregunta por su estado de ánimo.

Cada semana a las 8 de la tarde se le hacen las 4 preguntas que se encuentran marcadas sobre los efectos secundarios.

No se le realiza pregunta alguna sobre su actitud ante la medicación.

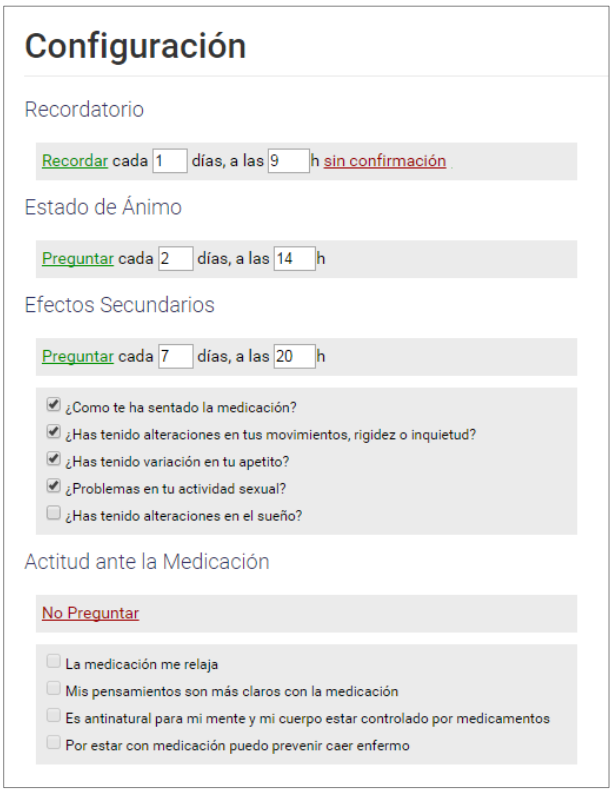

Figura 29. AppWeb - Configuración

Registros. Esta es la parte más extensa y en donde se representa de diversas maneras todos los datos de seguimiento del paciente.

La librería de Google Charts [52] dispone de multitud de gráficos que se han incorporado en la aplicación web para facilitar la interpretación y la evolución de los datos de seguimiento del paciente.

La Figura 30 muestra un ejemplo de las gráficas que representan los resultados de recordatorio con confirmación durante un periodo 62 días.

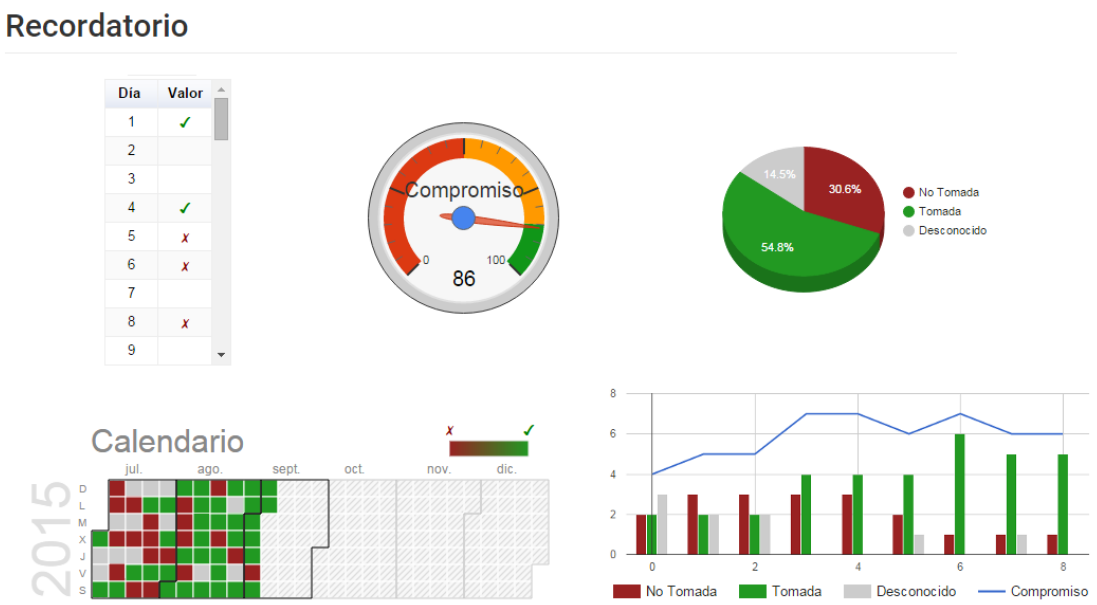

Figura 30. AppWeb – Datos de Seguimiento

Al igual que ocurre con los avisos de pánico, cuando se genera un nuevo dato de seguimiento, se actualiza la web sin necesidad de recargar la página.

Es interesante comprobar como la librería Meteor implementa esta característica. Para ello haremos uso de "Chrome Developer Tools" [53], una herramienta de análisis, medición y depuración de Google Chrome.

Al cargar la aplicación web, DevTools nos muestra los recursos descargados y su cronograma (explicaremos con más detalle este panel en el capítulo 5.1.3). Uno de esos recursos es un WebSocket, que permite establecer una comunicación bidireccional y persistente con el servidor. DevTools nos permite ver todos y cada uno de los mensajes que han sido transmitidos a través del WebSocket. La Figura 31 muestra un ejemplo de los mensajes necesarios para iniciar la comunicación y transmitir los datos de seguimiento iniciales.

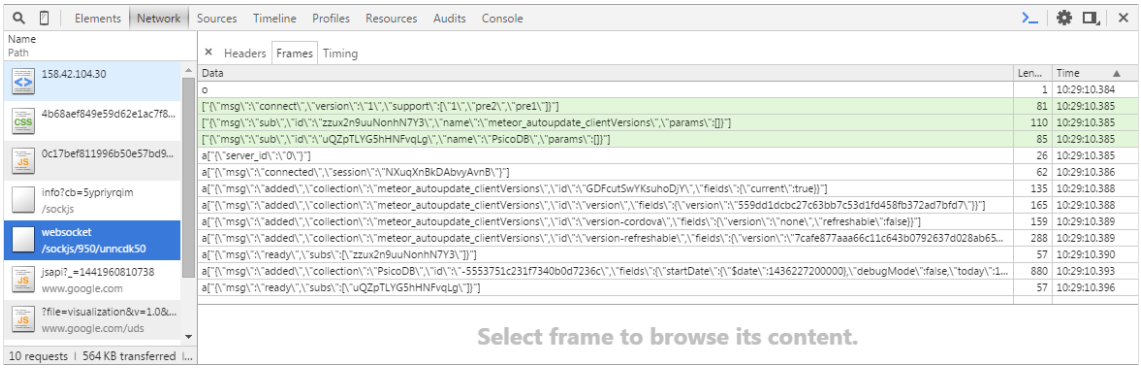

Figura 31. AppWeb – DevTools - Mensajes Generados y Recibidos por el WebSocket

Cuando un dato de seguimiento nuevo se genera y se introduce en la base de datos, el servidor web envía un mensaje al cliente a través del webSocket. Por ejemplo, el mensaje que representa un nuevo dato en el estado de ánimo sería el siguiente:

```
a["{\"msg\":\"changed\",\"collection\":\"PsicoDB\",\"id\":\"-
5553751c231f7340b0d7236c\",\"fields\":{\"records\":{\"feeling
\":[{\"day\":32,\"value\":5}]}}}"]
```

Este mensaje ocupa exactamente 156 bytes, lo que supone un exigencia en ancho de banda insignificante, y es lo único que necesita el cliente para actualizar la página web.

5. Resultados y Discusión

Se han realizado dos tipos de pruebas al sistema. El primer tipo está centrado en el análisis del rendimiento y fiabilidad del modelo y el prototipo. Hemos modificado la aplicación móvil para que sea capaz de enviar una gran cantidad de datos de seguimiento durante un largo periodo de tiempo. Hemos medido las prestaciones por parte del dispositivo móvil y del servidor web, tanto conectado por wifi como por red de datos móviles. También hemos medido las prestaciones a la hora de cargar la aplicación web en un navegador.

El segundo tipo de pruebas se ha realizado en un entorno real, donde dos médicos y un paciente psicótico del Departamento de Medicina del Hospital Clínico de Valencia han probado el sistema durante 2 meses.

5.1 Pruebas de Rendimiento

En estas pruebas, se ha puesto a prueba la fiabilidad y rendimiento del sistema, tanto de la aplicación móvil como del servidor web y de la aplicación web.

Para ello, se ha programado el envío consecutivo de 1000 datos de seguimiento, de tipo y valor aleatorios. Realizaremos las pruebas en un dispositivo móvil “Samsung Galaxy S2”, con procesador ARM Cortex-A9 de doble núcleo a 1.2 GHz, con 1GB de RAM.

En primer lugar, se ha realizado las pruebas conectados por wifi para conseguir realizar el número máximo de peticiones por segundo y cargar lo máximo posible el dispositivo móvil y el servidor web. Repetimos el proceso conectados a través de una red de datos móviles.

Para evaluar el rendimiento del dispositivo móvil, hemos hecho uso de las herramientas que dispone Android Studio para monitorizar el uso de la CPU y la RAM. Para evaluar el servidor web, hemos utilizado la herramienta disponible para Linux, *sysstat* [54], que monitoriza el estado de la máquina virtual, no del servidor, por lo que tendremos que detectar si algún otro proceso interfiere en nuestras mediciones. Por último, hemos evaluado el rendimiento de la aplicación web.

5.1.1 Dispositivo Móvil

Red Wifi

En una red wifi, el envío de los 1.000 datos ha tardado aproximadamente 72 segundos, eso supone una frecuencia de 14 envíos por segundos. Aunque pueda parecer poco, cabe recordar que cada envío supone la creación y destrucción de un nuevo servicio en el sistema operativo Android. Aunque esta aproximación no es la más apropiada cuando se realizan muchos envíos consecutivos, sí lo es para el caso de aplicación que nos ocupa, ya que implicará unos pocos envíos por equipo al día, y así nos aseguramos que el envío se efectúe aunque la actividad se destruya antes de enviar el dato.

Analizando el uso de la CPU del dispositivo móvil durante la ejecución, vemos en la Figura 32 que el uso medio durante la ejecución es del 20% con picos del 40%. Esto demuestra que la creación de servicios es algo costoso y que no debe realizarse de manera asidua.

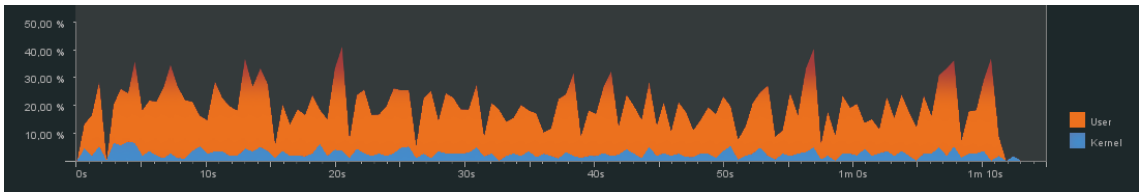

Figura 32. Uso de la CPU del dispositivo móvil. Conexión por wifi

Recordemos que este caso es una prueba de carga y nunca se dará en casos reales. La máxima frecuencia a la que se envía datos de seguimiento desde la aplicación móvil es de 4 envíos al día.

En cuanto al consumo de memoria RAM, podemos comprobar que esta se encuentra muy estable durante toda la ejecución, variando apenas un 12%.

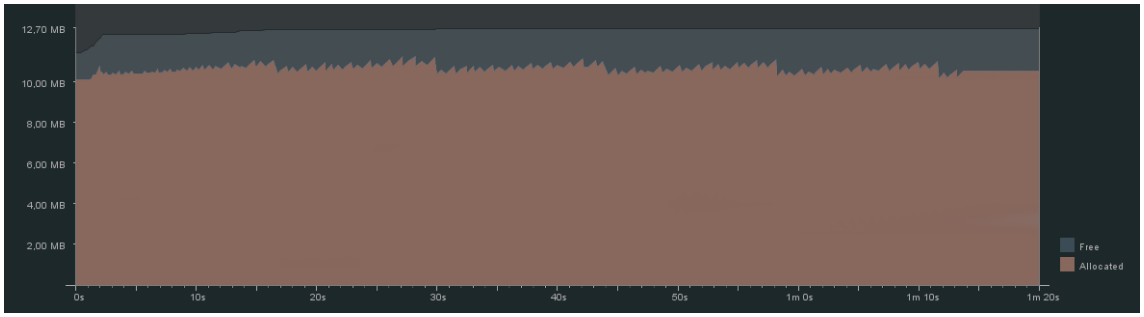

Figura 33. Uso de la Memoria RAM del dispositivo móvil. Conexión por wifi

Red de Datos Móviles

Cuando nos conectamos a una red de datos móviles de tipo HSDPA, las peticiones se han realizado a una frecuencia mucho menor. La ejecución del envío de 1.000 datos tarda 3 minutos y 52 segundos, unas 4.3 peticiones por segundo. Esto reduce el porcentaje de ocupación de la CPU, hasta un 14% de media con picos del 22%.

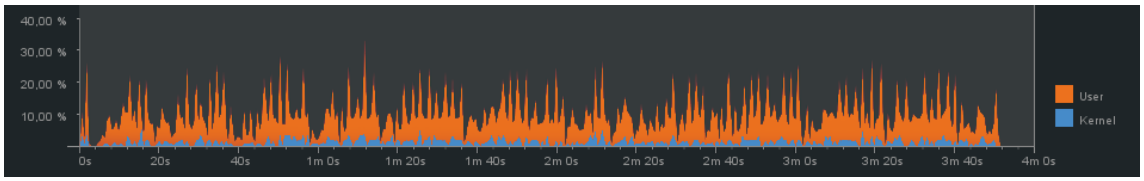

Figura 34. Uso de la CPU del dispositivo móvil. Conexión por Red de Datos Móviles

El consumo de memoria RAM no varía con respecto a las pruebas realizadas mediante wifi.

5.1.2 Servidor Web

Red Wifi

Recordemos que la herramienta utilizada de medición, sysstat [54], mide parámetros de la máquina virtual, no del servidor web, lo que significa que pueden aparecer interferencias en los resultados debidos a la existencia de otros procesos en ejecución en el sistema operativo.

Al medir la ocupación de la CPU durante la ejecución del envío masivo de datos de seguimiento, esta se mantiene entre el 6 y el 10% de manera estable.

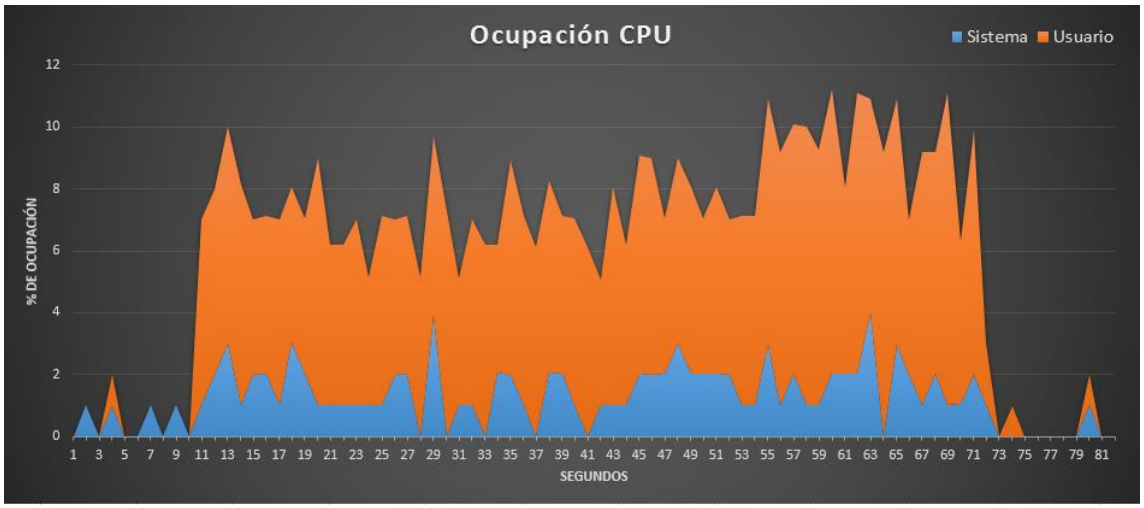

Figura 35. Uso de la CPU de la Máquina Virtual. Conexión por wifi

Si medimos la cantidad de paquetes recibidos por la máquina virtual, podemos comprobar que durante la ejecución, esta aumenta unos 40 paquetes por segundo de manera constante. Sin embargo, cada 30 segundos, la máquina recibe unos 50/70 paquetes adicionales, lo que hace suponer que existe otro una comunicación periódica generada por otro proceso ajeno a nuestro servidor web.

Figura 36. Paquetes Recibidos por la Máquina Virtual. Conexión por wifi

Red de Datos Móviles

Tal y como hemos mencionado anteriormente, al conectarnos por red de datos móviles, la frecuencia de envío de datos se reduce a 4.3 peticiones por segundos. Eso implica que el consumo de CPU caiga hasta el 3%.

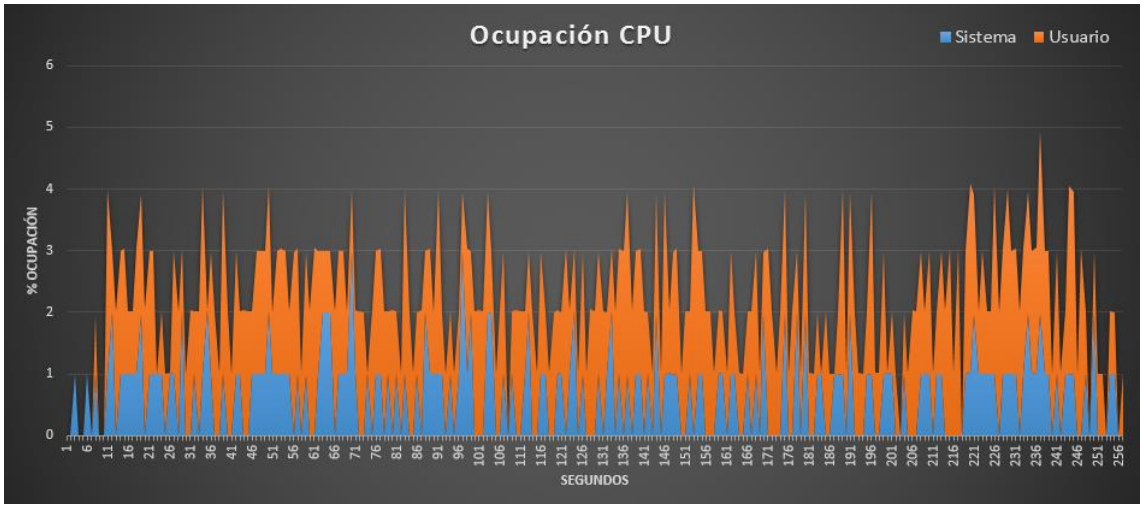

Figura 37. Uso de la CPU de la Máquina Virtual. Conexión por Red de Datos

De igual manera, la cantidad de paquetes recibidos por segundo por la máquina también resulta menor. Aquí se puede volver a comprobar la existencia de otro proceso en el sistema operativo que recibe unos 50/70 paquetes cada 30 segundos.

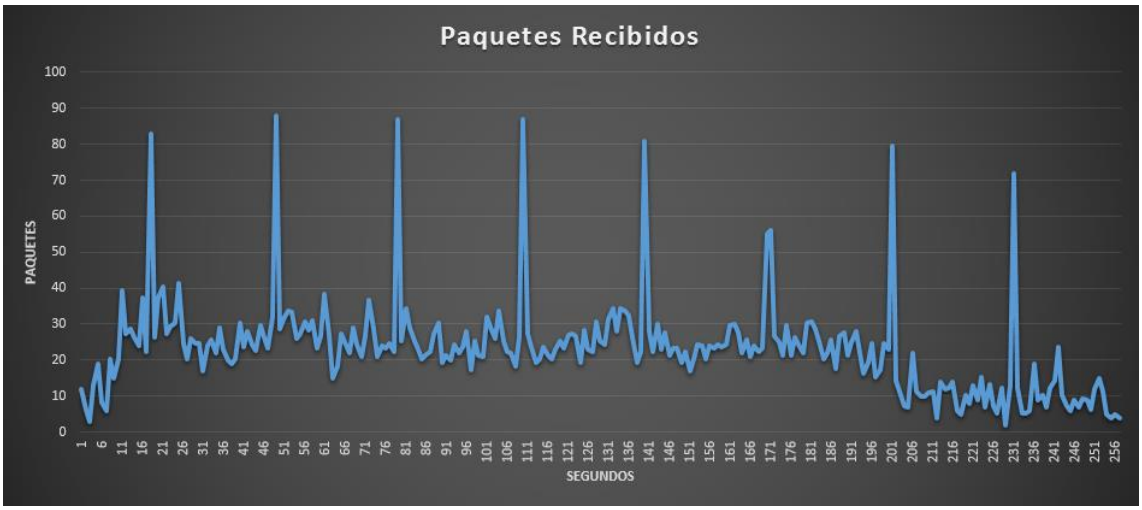

Figura 38. Paquetes Recibidos por la Máquina Virtual. Conexión por Red de Datos

5.1.3 Aplicación Web

Una vez los 1.000 datos de seguimiento han sido introducidos en la base de datos, debemos también evaluar las prestaciones de la aplicación web cargando tal cantidad de información y la cantidad de memoria RAM que ocupa.

Para ello hemos utilizado el navegador Google Chrome y su herramienta de monitorización y depuración, "Chrome Developer Tools" [53], para analizar la cantidad de ficheros descargados, sus tamaños y el tiempo requerido para procesarlos.

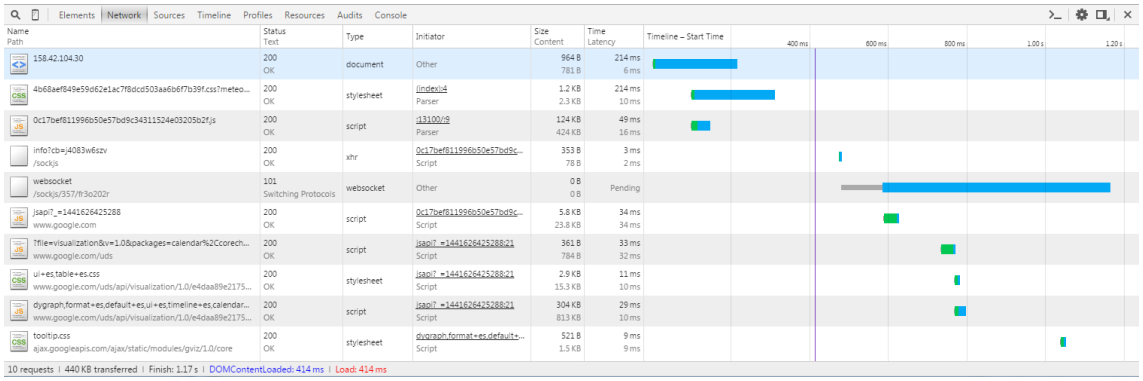

Figura 39. Chrome Developer Tools

En la Figura 39 se puede comprobar que la aplicación web se compone de un fichero .html de 781 bytes, un .css de 2.3KB y .js de 424KB (los tres primeros ficheros mostrados en el listado).

Estos ficheros han sido creados por Meteor, a partir de nuestros ficheros fuente. Cabe recordar que aunque nosotros teníamos una cantidad mayor de ficheros fuente, Meteor los agrupa en un solo fichero de cada tipo para minimizar el número de peticiones al servidor.

Una vez que el navegador haya *parseado* el .html y el .css, ejecute el fichero .js, y *renderice* el contenido de la web, esta ya puede ser mostrada al usuario. Esto ocurre exactamente a los 420ms (donde se sitúa la línea vertical morada). Sin embargo, en ese momento la aplicación no muestra ningún dato de la base de datos.

El siguiente paso consiste en iniciar un canal de comunicación bidireccional y persistente con el servidor mediante el protocolo DDP (un protocolo tipo *publish/subscribe* creado por Meteor) para obtener los datos de seguimiento existentes y recibir cualquier otro que pueda llegar posteriormente (cuarto y quinto fichero). Esta es la parte más costosa y dura unos 680ms.

Cuando llega el primer dato, se producen las últimas 5 peticiones para cargar la librería de Google Charts, que permite crear los gráficos.

En resumen, tal y como aparece en la barra de estado de la figura, se han realizado 10 peticiones, se han transferido un total de 440KB, y la aplicación web ha tardado un total de 1.17s en cargarse por completo.

Tras esto, podemos comprobar su ocupación en la memoria RAM mediante el Administrador de Tareas de Google Chrome.

En la Figura 40 se puede comprobar que el propio navegador consume 171MB, y que nuestra aplicación consume 71MB. Hemos incluido unas pocas pestañas más para poder establecer una comparación entre todas ellas. Estas son la página principal de la UPV (33MB) y de Google (49MB), la web de YouTube durante la reproducción de un video (82MB), y la bandeja de entrada de Gmail (146MB).

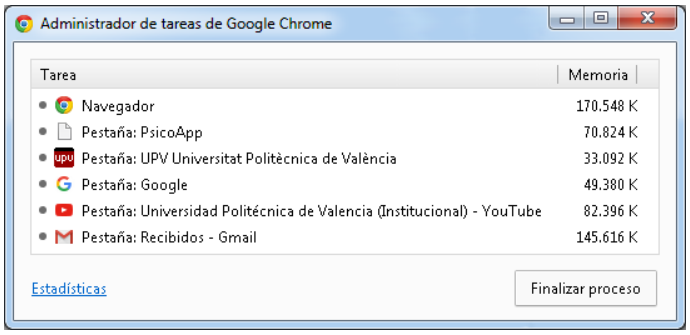

Figura 40. Administrador de Tareas de Google Chrome

Tras la realización de las pruebas en un entorno de alta demanda, hemos podido comprobar que el modelo y las tecnologías utilizadas son apropiados ya que el sistema ha mostrado ser fiable (no se ha perdido ningún dato de seguimiento) y eficiente (tiene unos consumos estables y predecibles, no llegando a saturar en ningún momento el dispositivo móvil ni el servidor web).

5.3 Pruebas en Entorno Real

Para las pruebas en entorno real, se ha contado con la colaboración de dos médicos y un paciente del Departamento de Medicina del Hospital Clínico de Valencia.

Las pruebas han consistido en utilizar la aplicación de manera continua durante 2 meses. Los dos médicos también han actuado como pacientes instalándose la aplicación móvil para poder evaluarla.

Durante esos 2 meses, Julio y Agosto, se ha producido un corte de luz programado durante 2 días en las instalaciones de la Universidad Politécnica de Valencia, lugar donde se alojaba el servidor web. Aunque la conexión a la aplicación web y la recolección de datos de seguimiento no ha sido posible durante ese tiempo, la aplicación móvil no ha sufrido ninguna excepción, ya que ha sido programada para que sea autónoma y tolerante a este tipo de situaciones.

Cuando los servidores han sido puestos en marcha de nuevo, no se ha requerido ninguna operación adicional en el dispositivo móvil para que el sistema volviese a funcionar.

Después de 2 meses de uso, se han introducido un total de 168 datos de seguimiento, no se ha producido ningún error en la aplicación móvil ni en el servidor web que comprometiese el progreso del sistema (a excepción del corte de luz).

6. Conclusiones y Trabajos Futuros

6.1 Conclusiones

El objetivo de este proyecto era evaluar y proponer un diseño de aplicación que facilite el seguimiento de la actividad y el estado de personas a través de dispositivos móviles y sus tecnologías.

De esta manera, se ha diseñado un sistema con el propósito de monitorizar el estado clínico y los progresos de un paciente para mejorar su adherencia al tratamiento.

Este sistema se compone de una base de datos NoSQL, MongoDB, que ha demostrado ser eficiente en su ejecución y flexible en su estructura de documentos, además de ser escalable y permitir ser fragmentada cuando exista la necesidad; un servidor web, Node.js, que permite satisfacer una alta demanda de peticiones sin colapsarse gracias a su arquitectura asíncrona orientada a eventos, y que su integración con MongoDB la hace especialmente eficiente; y una aplicación móvil basada en Android que permite interactuar con el usuario a través de notificaciones, que se mantiene activa en todo momento y que consume la mínima cantidad de recursos posible. También es destacable el *framework* utilizado para desarrollo de la aplicación web, Meteor, ya que incorpora mecanismos de sincronización entre cliente y servidor y está integrado con las tecnologías de Node.js y MongoDB.

6.2 Trabajos Futuros

Actualmente, y con la base de este trabajo, se está preparando una publicación conjunta una vez se hayan realizado experimentos con una versión del sistema que permita múltiples usuarios al mismo tiempo.

Por otra parte, se han identificado una serie de mejoras que se pretenden implementar:

- Actualmente, el sistema solo acepta un médico y un paciente. Una mejora indispensable será incorporar un sistema de cuentas de usuario que permita ingresar al sistema varios médicos y pacientes, al igual que algún mecanismo que permita asociar un médico a varios pacientes para que este solo pueda ver los progresos de sus pacientes y no del resto.
- Cuando un paciente envía un dato de seguimiento y no dispone de conexión a Internet o el servidor no es capaz de recibir el dato, este se pierde sin ser entregado. Será necesario que la aplicación móvil sea consciente de la situación, guarde el dato y reintente el envío de manera periódica o cuando se restablezca la conexión, incluso cuando el móvil se apague.
- Cuando un paciente pulsa el botón de pánico, el médico no se percata del evento hasta que entra en la aplicación web. Para agilizar esta situación, se podría implementar otra aplicación móvil que se instalaría en el dispositivo móvil de médico y que le notificaría cada vez que se produce este tipo de evento, para así darle la oportunidad de llamar al paciente por teléfono en ese mismo instante. Esto haría al sistema todavía más distribuido.

- La base de datos y el servidor web residen en la misma máquina y no se encuentran replicados. Actualmente, y como se ha visto en las pruebas de carga, no es necesario distribuir el sistema, pero si en un futuro la cantidad de usuarios aumenta considerablemente, esta configuración puede suponer un problema de rendimiento. Sería adecuado estudiar si llegados a ese punto, convendría separarlos en máquinas diferentes, replicarlos o fragmentar la base de datos para distribuir la carga y evitar cuellos de botella.
- El protocolo DDP implementado en Meteor nos facilita enormemente la sincronización de datos entre cliente web y servidor. Sin embargo, no se ha publicado todavía una versión oficial para Android. Sería conveniente estudiar las diversas versiones no oficiales y comprobar si su utilización vale la pena [55] [56].
- La monitorización de CPU y paquetes recibidos durante las pruebas de rendimiento en el servidor web han sido medidas para toda la máquina, lo que puede desvirtuar los resultados si aparecen interferencias de otros procesos del sistema. Existen aplicaciones para medir exclusivamente las prestaciones del servidor Node.js, como Keymetrics [58]; y de la aplicación Meteor, como Kadira [59].

7. Referencias

1. Valenstein, M., Blow. FC., Copeland, LA., McCarthy, JF., Zeber, JE., Gillon, L., Bingham, CR., Stavenger, T. (2004) Poor antipsychotic adherence among patients with chizophrenia: medication and patient factors. Schizophrenia Bulletin. 30(2): 255-264.

2. Cramer, JA., Rosenheck, R. (1998) Compliance with medication regimens for mental and physical disorders. Psychiatric services.49:196-201.

3. Lieberman, JA, et al (2005) Clinical Antipsychotic Trials of Intervention Effectiveness (CATIE) Investigators. Effectiveness of antipsychotic drugs in patients with chronic chizophrenia. New England Journal of Medicine; 353 (12): 1209-23.

4. Acosta FJ, et al (2012) Medication adherence in schizophrenia. World Journal of Psychiatry. 2(5): 74-82.

5. ITU (2015) United Nations specialized agency for information and communication technologies (ICTs): http://itu.int

6. World Health Organization (2011). mHealth - New horizons for health through mobile technologies. Based on the findings of the second global survey on eHealth. Global Observatory for eHealth series- Volume 3.

7. Bupa Health Pulse Survey (2010) Health and disease: https://bupa.com/healthpulse

8. Joseph, C., Kvedar, MD., Thomas Nesbitt, MD., MPH., Jule Kvedar, and Adam Darkins, MD. (2011). Journal of General Internal Medicine 26 (2):636-8 Society of General Internal Medicine (JGIM).

9. Buckee, CO., Wesolowski, A., Eagle, NN., Hansen, E., Snow, RW. (2013) Mobile phones and malaria: modeling human and parasite travel. Travel medicine and infectious disease; 11(1):15-22.

10. Danes, M., Whinder, F. (2013) Diabetes management goes digital. The Lancet. Diabetes & endocrinology; 1(1):17-8.

11. Kumar, N., Khunger, M., Gupta, A., Garg, N. (2015) A content analysis of smartphone-based application for hypertension management. Journal of the American Society of Hypertension; 9(2):130-6

12. Van Velthoven, MH., Brusamento, S., Majeed, A., Car, J. (2013) Scope and effectiveness of mobile phone messaging for HIV/AIDS care: a systematic review. Psychology, health & medicine; 18(2):182-202.

13. Bernhardt, JM., Usdan, S., Mays, D., Martin, R., Cremeens, J., Arriola, KJ., (2009). Alcohol assessment among college students using wireless mobile technology. Journal of Studies on Alcohol and Drugs. (70):771-5.

14. Freedman, MJ., Lester, KM., McNamara, C., Milby, JB., Schumacher, JE. (2006) Cell phones for ecological momentary assessment with cocaine-addicted homeless patients in treatment. Journal of Substance Abuse Treatment.30:105-11.

15. Obermayer, JL., Riley, WT., Asif, O., Jean-Mary, J. (2004) College smoking-cessation using cell phone text messaging. Journal of American college health. 53(2):71-8.

16. Depp, CA., Mausbach, B., Granholm, E., Cardenas, V., Ben-Zeev, D., Patterson, TL., Lebowitz, BD., Jeste, DV. (2010) Mobile interventions for severe mental illness: design and preliminary data from three approaches. The Journal of Nervous and Mental Disease. 198(10):715-21.

17. Baños, RM., Cebolla, A., Botella, C., García-Palacios, A., Oliver, E., Zaragoza, I., Alcaniz, M. (2011) Improving Childhood Obesity Treatment Using New Technologies: The ETIOBE System. Clinical practice and epidemiology in mental health. 4(7):62-6.

18. Baños, RM., Espinoza, M., García-Palacios, A., Cervera, JM., Esquerdo, G, Barrajón, E., Botella, C. (2013) A positive psychological intervention using virtual reality for patients with advanced cancer in a hospital setting: a pilot study to assess feasibility. Supportive care in cancer: Official Journal of The Multinational Association of Supportive Care in Cancer. 21(1):263-70.

19. Botella, C., Breton- López, J., Quero, S., Baños, RM., García- Palacios, A., Zaragoza, I., Alcaniz, M. (2011) Treating cockroach phobia using a serious game on a mobile phone and augmented reality exposure: A single case study. Computers in Human Behavior. 27(1) 217-227.

20. Garcia-Palacios, A., Herrero, R., Belmonte, MA., Castilla, D., Guixeres, J., Molinari, G., Baños, RM.(2014). Ecological momentary assessment for chronic pain in fibromyalgia using a smartphone: a randomized crossover study. European journal of pain. 18(6):862-72.

21. Kimhy, D., Myin-Germeys, I., Palmier-Claus, J., Swendsen, J. (2012) Mobile assessment guide for research in schizophrenia and severe mental disorders. Schizophrenia bulletin. 38(3):386-95.

22. Csikszentmihalyi M, Larson R. (1987) Validity and reliability of the Experience-Sampling Method. The Journal of Nervous and Mental Disease. 175(9):526-36.

23. Kimhy, D., Corcoran, C. (2008). Use of Palm computer as an adjunct to cognitive-behavioural therapy with an ultra-high-risk patient: a case report. Early Intervention in Psychiatry. 2(4):234-41.

24. Swendsen, J.,Ben-Zeev, D., Granholm, E. (2011).Real-time electronic ambulatory monitoring of substance use and symptom expression in schizophrenia. The American Journal of Psychiatry. 168(2):202-9.

25. Ennis, L., Rose, D., Denis, M., Pandit, N., Wykes, T. (2012) Can't surf, won't surf: the digital divide in mental health. Journal of mental health. 21(4):395-403.

26. Nosè, M., Barbui C. (2003) Systemic review of clinical interventions for reducing treatment non-adherence in psychosis. Epidemiologia e psichiatria sociale 12(4):272-86.

27. Haynes, RB., Ackloo, E., Sahota, N., McDonald, HP., Yao, X., (2008) Interventions for enhancing medication adherence. The Cochrane database of systematic reviews. 16;(2):CD000011.

28. Bartholomew, D. (2010). SQL vs. NoSQL: http://www.linuxjournal.com/article/10770

29. MongoDB: https://www.mongodb.org
30. Android: https://www.android.com
31. Android Developers: http://developer.android.com/index.html
32. Node.js: https://nodejs.org
33. Abernethy, M. (2011). ¿Qué es Node.js?: http://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs
34. Meteor: https://www.meteor.com
35. Berube, D. (2013). Desarrollo de sitios web fácil y en tiempo real con Meteor: http://www.ibm.com/developerworks/ssa/library/wa-meteor
36. DDP Specification: https://github.com/meteor/meteor/blob/devel/packages/ddp/DDP.md
37. Eugster, P., Felber, P. , Guerraoui, R. & Kermarrec, A. (2003). The many faces of publish/subscribe. ACM Computing Surveys. Volume 35 Issue 2: http://dl.acm.org/citation.cfm?id=857078
38. HTML - Mozilla Developer Network: https://developer.mozilla.org/es/docs/Web/HTML
39. HTML5 Specification - W3C: http://www.w3.org/TR/html
40. CSS - W3C: http://www.w3.org/Style/CSS
41. CSS - Mozilla Developer Network: https://developer.mozilla.org/es/docs/Web/CSS
42. Javascript - Mozilla Developer Network: https://developer.mozilla.org/es/docs/Web/JavaScript
43. jQuery: https://jquery.com
44. Android Studio: https://developer.android.com/sdk/index.html
45. Genymotion: https://www.genymotion.com
46. Robomongo: http://robomongo.org
47. Google Chrome: https://www.google.es/chrome/browser/desktop/index.html
48. Mozilla Firefox: https://www.mozilla.org/es-ES/firefox/new
49. Microsoft Internet Explorer: http://windows.microsoft.com/es-es/internet-explorer/download-ie
50. Opera: http://www.opera.com/es
51. Maxthon: http://es.maxthon.com
52. Google Charts: https://developers.google.com/chart/?hl=es
53. Chrome DevTools: https://developer.chrome.com/devtools
54. Sysstat: http://sebastien.godard.pagesperso-orange.fr
55. Android-DDP - GitHub: https://github.com/delight-im/Android-DDP

56. Meteor.js Android DDP Client - GitHub: https://github.com/kenyee/android-ddp-client
57. D3: http://d3js.org
58. Keymetrics: https://keymetrics.io
59. Kadira: https://kadira.io